```
IIIIIIIII       OOOOOOOOO       SSSSSSSSSSS  UUU         UUU  PPPPPPPPPPP
IIIIIIIII       OOOOOOOOO       SSSSSSSSSSS  UUU         UUU  PPPPPPPPPPP
IIIIIIIII       OOOOOOOOO       SSSSSSSSSSS  UUU         UUU  PPPPPPPPPPP
   III         OOO     OOO     SSS           UUU         UUU  PPP        PPP
   III         OOO     OOO     SSS           UUU         UUU  PPP        PPP
   III         OOO     OOO     SSS           UUU         UUU  PPP        PPP
   III         OOO     OOO     SSS           UUU         UUU  PPP        PPP
   III         OOO     OOO     SSS           UUU         UUU  PPP        PPP
   III         OOO     OOO     SSS           UUU         UUU  PPP        PPP
   III         OOO     OOO      SSSSSSSSS    UUU         UUU  PPPPPPPPPPP
   III         OOO     OOO      SSSSSSSSS    UUU         UUU  PPPPPPPPPPP
   III         OOO     OOO      SSSSSSSSS    UUU         UUU  PPPPPPPPPPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
   III         OOO     OOO              SSS  UUU         UUU  PPP
IIIIIIIII       OOOOOOOOO       SSSSSSSSSSS  UUUUUUUUUUUUUUU  PPP
IIIIIIIII       OOOOOOOOO       SSSSSSSSSSS  UUUUUUUUUUUUUUU  PPP
IIIIIIIII       OOOOOOOOO       SSSSSSSSSSS  UUUUUUUUUUUUUUU  PPP
```

FILEID**DRSUP

```
DDDDDDDD    RRRRRRRR      SSSSSSSS  UU      UU  PPPPPPPP
DDDDDDDD    RRRRRRRR      SSSSSSSS  UU      UU  PPPPPPPP
DD      DD  RR      RR  SS            UU      UU  PP      PP
DD      DD  RR      RR  SS            UU      UU  PP      PP
DD      DD  RR      RR  SS            UU      UU  PP      PP
DD      DD  RR      RR  SS            UU      UU  PP      PP
DD      DD  RRRRRRRR      SSSSSS      UU      UU  PPPPPPPP
DD      DD  RRRRRRRR      SSSSSS      UU      UU  PPPPPPPP
DD      DD  RR  RR                SS  UU      UU  PP
DD      DD  RR  RR                SS  UU      UU  PP
DD      DD  RR    RR              SS  UU      UU  PP              ....
DD      DD  RR    RR              SS  UU      UU  PP              ....
DDDDDDDD    RR      RR  SSSSSSSS      UUUUUUUUUU  PP              ....
DDDDDDDD    RR      RR  SSSSSSSS      UUUUUUUUUU  PP              ....


LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  XF$DRSUP -- DR32 SUPPORT ROUTINES
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;****************************************************************************
0000     6  ;*                                                                          *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                    *
0000    10  ;*                                                                          *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16  ;*  TRANSFERRED.                                                            *
0000    17  ;*                                                                          *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                            *
0000    21  ;*                                                                          *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    24  ;*                                                                          *
0000    25  ;*                                                                          *
0000    26  ;****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:     DR32 SUPPORT ROUTINES
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;       Provide high-level language interface to DR32
0000    34  ;
0000    35  ; ENVIRONMENT:  USER MODE LIBRARY ROUTINES
0000    36  ;
0000    37  ; MODIFIED BY:
0000    38  ;
0000    39  ;       V03-003 TCM0004         Trudy C. Matthews       30-Mar-1983
0000    40  ;               Correct two bugs introduced in TCM0003 that could cause
0000    41  ;               user-specified action routines to not be called.
0000    42  ;
0000    43  ;       V03-002 TCM0003         Trudy C. Matthews       18-Jun-1982
0000    44  ;               Change XF$STARTDEV so that it sets the GO bit before exiting.
0000    45  ;
0000    46  ;               Correct two problems in XF$PKTBLD -- (1) if an action routine
0000    47  ;               was not specified, the packet would always be inserted at the
0000    48  ;               tail of the queue (even if MODES specified "insert at head");
0000    49  ;               (2) if MODES was defaulted and an action routine specified,
0000    50  ;               an access violation would occur.
0000    51  ;
0000    52  ;       V03-001 SBL3001         Steven B. Lionel        30-Mar-1982
0000    53  ;               Change module name to XF$DRSUP.  Make PRE_AST, GET_ADDR
0000    54  ;               and DEVICE_FAB local symbols.
0000    55  ;
0000    56  ;       V02-004 PRD0006         Paul R. DeStefano       1-Mar-1982
0000    57  ;               Correct symbols LIB$GET_VM in ALOCCMD and LIB$FREE_VM in
```

```
0000     58 ;                  XF$CLEANUP.  Symbols were not position independent.
0000     59 ;
0000     60 ;       V02-003 TCM0002         Trudy C. Matthews       6-Jul-1981
0000     61 ;               In XF$GETPKT, correct the instruction that stores the
0000     62 ;               function code in the user supplied argument to only store
0000     63 ;               a word instead of a longword.
0000     64 ;
0000     65 ;       V02-002 TCM0001         Trudy C. Matthews       15-Jun-1981
0000     66 ;               In ALOCCMD, correct algorithm that initializes free command
0000     67 ;               memory pointers.
0000     68 ;
0000     69 ;--
```

```
0000     71          .SBTTL  DECLARATIONS
0000     72 ;
0000     73 ; MACROS:
0000     74 ;
0000     75          $SSDEF                          ;define status returns
0000     76          $XFDEF                          ;DR32-specific definitions
0000     77          $$DRDEFS                        ;support routine definitions
0000     78          $CTXDEF                         ;offsets into contxt array
0000     79          $IODEF                          ;IO status definitions
0000     80          $SHRDEF                         ;shared status definitions
0000     81
0000     82 ;macro DEFAULT_TEST tests for defaulted FORTRAN-procedure arguments
0000     83
0000     84          .MACRO  DEFAULT_TEST    ARGPOS, LABEL1, LABEL2
0000     85 ;ARGPOS contains the position of an argument in the argument list
0000     86
0000     87          CMPL    (AP), #ARGPOS           ;arg given?
0000     88          BLSS    LABEL1                  ;argument was not supplied
0000     89          TSTL    <ARGPOS*4>(AP)          ;if address = 0
0000     90          BEQL    LABEL2                  ;argument was defaulted
0000     91          .ENDM   DEFAULT_TEST
0000     92
0000     93 ;macro QRETRY executes an interlocked queue instruction and retries
0000     94 ;if failure.
0000     95 ;INPUTS:
0000     96 ;        OPCODE = opcode name: INSQHI, INSQTI, REMQHI, REMQTI
0000     97 ;        OPERAND1 = first operand for opcode
0000     98 ;        OPERAND2 = second operand for opcode
0000     99 ;        SUCCESS = label to branch to if operation succeeds
0000    100 ;        ERROR = label to branch to if operation fails
0000    101 ;OUTPUTS:
0000    102 ;        R0 is destroyed
0000    103
0000    104          .MACRO  QRETRY  OPCODE,OPERAND1,OPERAND2,SUCCESS,ERROR,?LOOP,?OK
0000    105          CLRL    R0
0000    106 LOOP:
0000    107          OPCODE  OPERAND1, OPERAND2
0000    108          .IF NB  SUCCESS                 ;"C" bit clear <=> success
0000    109          BCC     SUCCESS
0000    110          .IFF
0000    111          BCC     OK
0000    112          .ENDC
0000    113          AOBLSS  #RETRY_LIMIT, R0, LOOP  ;queue is interlocked. Retry.
0000    114          .IF NB ERROR
0000    115          BRB     ERROR                   ;retry limit exceeded and queue
0000    116          .ENDC                           ;is still locked. Assume error.
0000    117 OK:
0000    118          .ENDM   QRETRY
0000    119
0000    120 ;
0000    121 ; REGISTER CONVENTIONS:
0000    122 ;        R6 : address of CONTXT array
0000    123 ;        R7 : address of current command packet
0000    124 ;        R10: address of command block
0000    125 ;
```

```
                        .SBTTL  XF$SETUP
                ;++
                ;CONTXT ARRAY:
                ;
                ;        31                              0
                ;       +--------------------------------+
                ;       |                                |   :CTX$Q_IOSB
                ;       +--    I/O status block        --+
                ;       |                                |
                ;       +--------------------------------+
                ;       |   device and command control   |   :CTX$L_CONTROL
                ;       +--------------------------------+
                ;       |        buffer size             |   :CTX$L_BYTECNT
                ;       +--------------------------------+
                ;       |        buffer address          |   :CTX$L_BFRVA
                ;       +--------------------------------+
                ;       |  residual memory byte count    |   :CTX$L_MEMCNT
                ;       +--------------------------------+
                ;       |   residual DDI byte count      |   :CTX$L_DDICNT
                ;       +--------------------------------+
                ;       |  DR32 status longword (DSL)    |   :CTX$L_DSL
                ;       +--------------------------------+
                ;       |     size of command block      |   :CTX$L_CMDSIZ   :CTX$B_CMDTBL
                ;       +--------------------------------+
                ;       |   address of command block     |   :CTX$L_CMDBLK
                ;       +--------------------------------+
                ;       |      size of data block        |   :CTX$L_DATASIZ
                ;       +--------------------------------+
                ;       |     address of data block      |   :CTX$L_DATABLK
                ;       +--------------------------------+
                ;       |  address of pre- AST routine   |   :CTX$L_PRE_AST
                ;       +--------------------------------+
                ;       |      pre- AST parameter        |   :CTX$L_PRE_PARM
                ;       +--------------------------------+
                ;       |                | flags | datart|   :CTX$B_DATART  CTX$B_FLAGS
                ;       +--------------------------------+
                ;       | addr to receive addr of gobit  |   :CTX$L_GOBITADR
                ;       +--------------------------------+
                ;       | event flag #   | # of buffers  |   :CTX$W_NUMBUF   :CTX$W_EFN
                ;       +--------------------------------+
                ;       | address of packet AST routine  |   :CTX$L_PKTAST
                ;       +--------------------------------+
                ;       |    packet AST parameter        |   :CTX$L_ASTPARM
                ;       +--------------------------------+
                ;       | size of each buffer in BARRAY  |   :CTX$L_BUFSIZ
                ;       +--------------------------------+
                ;       |    address of IDEVMSG array    |   :CTX$L_IDEVMSG
                ;       +--------------------------------+
                ;       |    address of ILOGMSG array    |   :CTX$L_ILOGMSG
                ;       +--------------------------------+
                ;       |size of IDEVMSG|size of ILOGMSG |   :CTX$W_ILOGSIZ :CTX$W_IDEVSIZ
                ;       +--------------------------------+
                ;       | address of free memory list    |   :CTX$L_FREELIST
                ;       +--------------------------------+
                ;               note: CONTXT offsets are defined in $CTXDEF
```

```
                     0000   183 ; FUNCTIONAL DESCRIPTION:
                     0000   184 ;
                     0000   185 ;       (1) allocates command area
                     0000   186 ;       (2) allocates and initializes hardware queue headers
                     0000   187 ;       (3) initializes free command memory list
                     0000   188 ;       (4) initializes many fields in the CONTXT array
                     0000   189 ;
                     0000   190 ; CALLING SEQUENCE:
                     0000   191 ;
                     0000   192 ;       CALLS/G XF$SETUP (contxt, barray, bufsiz, numbuf, [idevmsg], -
                     0000   193 ;                        [idevsiz], [ilogmsg], [ilogsiz], [cmdsiz], -
                     0000   194 ;                        [status])
                     0000   195 ;
                     0000   196 ; INPUT PARAMETERS:
                     0000   197 ;    offsets to AP:
00000004             0000   198 ;       CONTXT = 4      ;a  50-word array that contains context and
                     0000   199 ;                       ;status information concerning the current
                     0000   200 ;                       ;transfer
00000008             0000   201 ;       BARRAY = 8      ;base address of data area
0000000C             0000   202 ;       BUFSIZ = 12     ;the size in bytes of each buffer in BARRAY
00000010             0000   203 ;       NUMBUF = 16     ;the number of buffers in BARRAY
00000014             0000   204 ;       IDEVMSG = 20    ;array to receive input device messages
00000018             0000   205 ;       IDEVSIZ = 24    ;size in bytes of device message array
0000001C             0000   206 ;       ILOGMSG = 28    ;array to receive input log messages
00000020             0000   207 ;       ILOGSIZ = 32    ;size in bytes of log message array
00000024             0000   208 ;       CMDSIZ = 36     ;size of command area to allocate
                     0000   209 ;
                     0000   210 ; OUTPUT PARAMETERS:
                     0000   211 ;
00000028             0000   212 ;       STATUS = 40     ;a longword array to receive status of call
                     0000   213 ;
                     0000   214 ; IMPLICIT OUTPUTS:
                     0000   215 ;
                     0000   216 ;       fields in CONTXT:       CTX$L_BUFSIZ
                     0000   217 ;                               CTX$L_CMDBLK
                     0000   218 ;                               CTX$L_CMDSIZ
                     0000   219 ;                               CTX$L_DATABLK
                     0000   220 ;                               CTX$L_DATASIZ
                     0000   221 ;                               CTX$L_IDEVMSG
                     0000   222 ;                               CTX$W_IDEVSIZ
                     0000   223 ;                               CTX$L_ILOGMSG
                     0000   224 ;                               CTX$W_ILOGSIZ
                     0000   225 ;                               CTX$W_NUMBUF
                     0000   226 ;
                     0000   227 ; COMPLETION CODES:
                     0000   228 ;
                     0000   229 ;       (1) SS$_NORMAL          normal successful completion
                     0000   230 ;       (2) SS$_BADPARAM        invalid input argument
                     0000   231 ;       (3) error status returns from LIB$GET_VM
                     0000   232 ;
                     0000   233 ; SIDE EFFECTS:
                     0000   234 ;
                     0000   235 ;       NONE
                     0000   236 ;
                     0000   237 ;--
```

```
                        00000000   239          .PSECT  _XF$CODE        SHR,PIC,EXE,NOWRT
                              0000 240
                        004C  0000 241          .ENTRY  XF$SETUP        ^M<R2,R3,R6>
                              0002 242
                              0002 243 ;store input parameters in CONTXT array
                              0002 244
                   50   14   3C   0002 245          MOVZWL  #SS$_BADPARAM, R0       ;set R0 for possible error
                              0005 246                                          ;return
                   04   6C   D1   0005 247          CMPL    (AP), #4                ;4 obligatory parameters
                        03   18   0008 248          BGEQ    10$
                     008C   31   000A 249          BRW     FINISH                  ;needed parameter defaulted
              56   04 AC   D0   000D 250 10$:       MOVL    CONTXT(AP), R6          ;R6 contains address of CONTXT
                              0011 251                                          ;array
          2C A6    08 AC   D0   0011 252          MOVL    BARRAY(AP), CTX$L_DATABLK(R6)   ;address of buf array
          40 A6    10 BC   B0   0016 253          MOVW    @NUMBUF(AP), CTX$W_NUMBUF(R6)   ;number of buffers
          4C A6    0C BC   D0   001B 254          MOVL    @BUFSIZ(AP), CTX$L_BUFSIZ(R6)   ;size of each buffer
                              0020 255
                              0020 256 ;determine size of data area, and store in CONTXT
                              0020 257
                   52   40 A6   3C   0020 258          MOVZWL  CTX$W_NUMBUF(R6), R2    ;R2 <- # of buffers in BARRAY
       28 A6       4C A6   52   C5   0024 259          MULL3    R2, =                          ;number of buffers X
                              002A 260                  CTX$L_BUFSIZ(R6), -            ;size of each buffer
                              002A 261                  CTX$L_DATASIZ(R6)
                              002A 262
                              002A 263 ;store addresses and sizes of arrays to receive input messages
                              002A 264
                              002A 265 MSG_ARRAYS:
                   50 A6    7C   002A 266          CLRQ    CTX$L_IDEVMSG(R6)       ;zero addresses of device and
                              002D 267                                          ;log message arrays
                   58 A6    D4   002D 268          CLRL    CTX$W_ILOGSIZ(R6)       ;assume sizes of device and
                              0030 269                                          ;log message arrays = 0
                              0030 270
                              0030 271          DEFAULT_TEST    <IDEVMSG/4>, 10$, 10$
                              003A 272                                          ;if IDEVMSG defaulted, goto 10$
          50 A6    14 AC   D0   003A 273          MOVL    IDEVMSG(AP), CTX$L_IDEVMSG(R6)
                              003F 274                                          ;store addr of IDEVMSG array
                              003F 275          DEFAULT_TEST    <IDEVSIZ/4>, 10$, 10$
                              0049 276                                          ;if IDEVSIZ defaulted, goto 10$
          5A A6    18 BC   B0   0049 277          MOVW    @IDEVSIZ(AP), CTX$W_IDEVSIZ(R6)
                              004E 278                                          ;store size of IDEVMSG array
                              004E 279 10$:     DEFAULT_TEST    <ILOGMSG/4>, CMDSIZ_TEST, CMDSIZ_TEST
                              0058 280                                          ;if ILOGMSG defaulted, goto CMDSIZ_TEST
          54 A6    1C AC   D0   0058 281          MOVL    ILOGMSG(AP), CTX$L_ILOGMSG(R6)
                              005D 282                                          ;store addr of ILOGMSG array
                              005D 283          DEFAULT_TEST    <ILOGSIZ/4>, CMDSIZ_TEST, CMDSIZ_TEST
                              0067 284                                          ;if ILOGSIZ defaulted, goto CMDSIZ_TEST
          58 A6    20 BC   B0   0067 285          MOVW    @ILOGSIZ(AP), CTX$W_ILOGSIZ(R6)
                              006C 286                                          ;store size of ILOGSIZ array
                              006C 287
```

```
                        006C   289 CMDSIZ_TEST:
                        006C   290 ;determine size of command area , and store in CONTXT
                        006C   291
                        006C   292         DEFAULT_TEST    <CMDSIZ/4>, COMSIZ, COMSIZ
                        0076   293                                         ;was size of command block given
                        0076   294                                         ;if not, goto COMSIZ
                        0076   295
20 A6   24 BC   18   C1 0076   296         ADDL3   #24, @CMDSIZ(AP), CTX$L_CMDSIZ(R6)      ;yes, add space
                        007C   297                                         ;for queue headers,
                        007C   298                                         ;and store in CONTXT
                   14   11 007C   299         BRB     ALOC
                        007E   300
                        007E   301 ;default command size = NUMBUF * (size of fixed portion of command
                        007E   302 ;packet + idevsiz + ilogsiz) * arbitrary constant ( originally = 3)
                        007E   303
                        007E   304 COMSIZ:
                   53   D4 007E   305         CLRL    R3
53   58 A6    5A A6   A1 0080   306         ADDW3   CTX$W_IDEVSIZ(R6), -    ;this sum will be <= 256
                        0086   307                 CTX$W_ILOGSIZ(R6), R3
              53   20   C0 0086   308         ADDL2   #XF$B_PKT_DEVMSG, R3    ;add in fixed portion of packet
   20 A6      53   52   C5 0089   309         MULL3   R2, R3, CTX$L_CMDSIZ(R6) ;R2 = NUMBUF
         20 A6   03   C4 008E   310         MULL2   #CMDSIZ_K, CTX$L_CMDSIZ(R6)     ;multiply by constant
                        0092   311
                        0092   312 ;intialize the addr of the addr of the go bit in CONTXT now so that
                        0092   313 ;XF$PKTBLD may be called before XF$STARTDEV.  It will be initialized
                        0092   314 ;again in XF$STARTDEV; this is a dummy initialization.
                        0092   315
                        0092   316 ALOC:
              39 A6   DE 0092   317         MOVAL   <CTX$B_CMDTBL + XF$B_CMT_FLAGS>(R6),-   ;request go bit
              3C A6      0095   318                 <CTX$B_CMDTBL + XF$L_CMT_GBITAD>(R6)    ;addr in here
                        0097   319
                        0097   320 ;All input parameters have been stored.  Now allocate and initialize
                        0097   321 ;command area.
                        0097   322
                   0F   10 0097   323         BSBB    ALOCCMD                 ;allocate command area
                        0099   324                                         ;and initialize queue heads
                        0099   325                                         ;status returned in R0
                        0099   326 FINISH:
                        0099   327         DEFAULT_TEST    <STATUS/4>, END, END    ;was status arg given?
                        00A3   328                                         ;if not, branch to END
28 BC   50   D0 00A3   329         MOVL    R0, @STATUS(AP)         ;yes, store status return
                        00A7   330
                   04   00A7   331 END:    RET
```

```
                        .SBTTL  ALOCCMD -- ALLOCATE COMMAND AREA
                                        AND INITIALIZE HARDWARE QUEUES
;
;++
; FUNCTIONAL DESCRIPTION:
;
;       This routine is called by XF$SETUP to dynamically allocate the
;       virtual memory that will be used as the command block during a
;       DR32 data transfer.  If successful, it initializes the first 3
;       quadwords as headers of the INPUT, TERMINATION, and FREE queues.
;       When this routine exits, command memory looks like:
;       31                      0
;       +-----------------------+
;       |     INPTQ FLINK       |  :CMD$L_INPTQ
;       +-----------------------+
;       |     INPTQ BLINK       |
;       +-----------------------+
;       |     TERMQ FLINK       |  :CMD$L_TERMQ
;       +-----------------------+
;       |     TERMQ BLINK       |
;       +-----------------------+
;       |     FREEQ FLINK       |  :CMD$L_FREEQ
;       +-----------------------+
;       |     FREEQ BLINK       |
;       +-----------------------+
;       |           0           |  :first longword of command space
;       +-----------------------+
;       | size of packet memory |
;       +-----------------------+
;       |                       |
;       |           .           |
;       |           .           |
;       |  memory available to  |
;       |   build command packets|
;       |           .           |
;       |           .           |
;       +-----------------------+
;
; CALLING SEQUENCE:
;
;       BSBB    ALOCCMD
;       BSBW    ALOCCMD
;                               called by XF$SETUP
;
```

```
                     ;  INPUT PARAMETERS:
                     ;
                     ;        NONE
                     ;
                     ;  IMPLICIT INPUTS:
                     ;
                     ;        address of CONTXT array in R6
                     ;        CONTXT fields used as inputs:    CTX$L_CMDSIZ
                     ;
                     ;  OUTPUT PARAMETERS:
                     ;
                     ;        NONE
                     ;
                     ;  IMPLICIT OUTPUTS:
                     ;
                     ;        fields in CONTXT:
                     ;                CTX$L_CMDBLK     address of allocated command area
                     ;                CTX$L_FREELIST   address of first longword on free list
                     ;
                     ;  COMPLETION CODES:
                     ;
                     ;        R0 contains status of call to LIB$GET_VM
                     ;
                     ;  SIDE EFFECTS:
                     ;
                     ;        NONE
                     ;
                     ;--
                     
```

```
                                00A8   406
                                00A8   407 ALOCCMD:
                          5A    DD 00A8   408         PUSHL   R10                     ;save register
                                00AA   409
                                00AA   410 ;round size of command area up to next page boundary before allocating
                                00AA   411
                20 A6   000001FF 8F   C0 00AA   412         ADDL2   #PAGEMASK, CTX$L_CMDSIZ(R6)     ;increase size past
                                00B2   413                                                         ;next boundary
                    20 A6   01FF 8F   AA 00B2   414         BICW    #PAGEMASK, CTX$L_CMDSIZ(R6)     ;truncate back to
                                00B8   415                                                         ;multuple
                                00B8   416
                                00B8   417 ;allocate command area
                                00B8   418
                          24 A6    DF 00B8   419         PUSHAL   CTX$L_CMDBLK(R6)               ;receives address of
                                00BB   420                                                         ;allocated area
                          20 A6    DF 00BB   421         PUSHAL  CTX$L_CMDSIZ(R6)                ;size to allocate
              00000000'GF    02    FB 00BE   422         CALLS   #2, G^LIB$GET_VM                ;get virtual memory
                          1A 50    E9 00C5   423         BLBC    R0, 10$                         ;error check
                                00C8   424
                                00C8   425 ;initialize hardware queues
                                00C8   426
                       5A 24 A6    D0 00C8   427         MOVL    CTX$L_CMDBLK(R6), R10           ;R10 points to beginning
                                00CC   428                                                         ;of command block
                          6A    7C 00CC   429         CLRQ    CMD$L_INPTQ(R10)                ;initialize queue head
                       08 AA    7C 00CE   430         CLRQ    CMD$L_TERMQ(R10)                ;initialize
                       10 AA    7C 00D1   431         CLRQ    CMD$L_FREEQ(R10)                ;initialize queue head
                                00D4   432
                                00D4   433 ;initialize list of free memory chunks
                                00D4   434
                       18 AA    DE 00D4   435         MOVAL   <CMD$L_FREEQ+8>(R10),-  ;FREELIST points to
                       5C A6           00D7   436                 CTX$L_FREELIST(R6)      ;first available blk of memory
                                00D9   437
                                00D9   438 ;The amount of command block memory available for building packets =
                                00D9   439 ;the size of command area - space reserved for queue heads.
                                00D9   440
          1C AA    20 A6    18    C3 00D9   441         SUBL3   #24,CTX$L_CMDSIZ(R6), - ;store size of initially
                                00DF   442                 <CMD$L_FREEQ+12>(R10)   ;available command memory
                          5C B6    D4 00DF   443         CLRL    @CTX$L_FREELIST(R6)     ;initialize free block pointer
                       0400 8F    BA 00E2   444 10$:    POPR    #^M<R10>
                                05    00E6   445         RSB
```

```
                     00E7   447          .SBTTL  XF$STARTDEV -- START DEVICE
                     00E7   448 ;++
                     00E7   449 ; FUNCTIONAL DESCRIPTION:
                     00E7   450 ;
                     00E7   451 ;       (1) build command table required by Startdata QIO
                     00E7   452 ;       (2) assign a channel to the device
                     00E7   453 ;       (3) issue the Startdata QIO
                     00E7   454 ;
                     00E7   455 ; CALLING SEQUENCE:
                     00E7   456 ;
                     00E7   457 ;       CALLS/G XF$STARTDEV (CONTXT, DEVNAM,  [PKTAST], [ASTPARM],
                     00E7   458 ;                            [EFN], [MODES], [DATART], [STATUS])
                     00E7   459 ;
                     00E7   460 ; INPUT PARAMETERS:
                     00E7   461 ;
                     00E7   462 ; offsets to AP:
00000004             00E7   463          CONTXT = 4              ;address of CONTXT array
00000008             00E7   464          DEVNAM = 8              ;character string; device name of DR32
0000000C             00E7   465          PKTAST = 12             ;address of packet AST
00000010             00E7   466          ASTPARM = 16            ;address of AST parameter
00000014             00E7   467          EFN = 20                ;event flag associated with transfer
00000018             00E7   468          MODES = 24              ;contains several switches
0000001C             00E7   469          DATART = 28             ;data rate of transfer
                     00E7   470 ;
                     00E7   471 ; IMPLICIT INPUTS:
                     00E7   472 ;
                     00E7   473 ;fields in the CONTXT array:
                     00E7   474 ;       CTX$L_CMDBLK
                     00E7   475 ;       CTX$L_CMDSIZ
                     00E7   476 ;       CTX$L_DATABLK
                     00E7   477 ;       CTX$L_DATASIZ
                     00E7   478 ;
                     00E7   479 ; OUTPUT PARAMETERS:
                     00E7   480 ;
00000020             00E7   481          STATUS = 32             ;optional status return
                     00E7   482 ;
                     00E7   483 ; IMPLICIT OUTPUTS:
                     00E7   484 ;
                     00E7   485 ;       various fields in the CONTXT array
                     00E7   486 ;
                     00E7   487 ; COMPLETION CODES:
                     00E7   488 ;
                     00E7   489 ;       (1) SS$_NORMAL          normal successful completion
                     00E7   490 ;       (2) SS$_BADPARAM        needed parameter defaulted
                     00E7   491 ;       (3) error returns from:
                     00E7   492 ;               $CREATE
                     00E7   493 ;               $QIO
                     00E7   494 ;
                     00E7   495 ; SIDE EFFECTS:
                     00E7   496 ;
                     00E7   497 ;       NONE
                     00E7   498 ;
                     00E7   499 ;--
```

```
                                00E7    501
                                00E7    502                .PSECT  _XF$DATA        NOEXE
                                0000    503 DEVICE_FAB:
                                0000    504                $FAB    FOP = UFO                       ;User File Open option
                                0050    505
                                0050    506
                                00E7    507                .PSECT  XF$CODE         EXE,NOWRT,SHR,PIC
                           004C 00E7    508                .ENTRY  XF$STARTDEV     ^M<R2,R3,R6>
                                00E9    509
                     56   04 AC   D0    00E9    510                MOVL    CONTXT(AP), R6          ;R6 <- addr of CONTXT
                                00ED    511
                                00ED    512 ;++
                                00ED    513 ;Two of the device-dependent parameters of the Startdata QIO are the
                                00ED    514 ;address and the size of a 'command table'.
                                00ED    515 ;The format of this command table is:
                                00ED    516 ;           31                              0
                                00ED    517 ;           +-------------------------------+
                                00ED    518 ;           |    size of command block      |   :XF$L_CMT_CBLKSIZ
                                00ED    519 ;           +-------------------------------+
                                00ED    520 ;           |   address of command block    |   :XF$L_CMT_CBLKAD
                                00ED    521 ;           +-------------------------------+
                                00ED    522 ;           |    size of data block         |   :XF$L_CMT_BBLKSIZ
                                00ED    523 ;           +-------------------------------+
                                00ED    524 ;           |    address of data block      |   :XF$L_CMT_BBLKAD
                                00ED    525 ;           +-------------------------------+
                                00ED    526 ;           | address of packet AST routine |   :XF$L_CMT_PASTAD
                                00ED    527 ;           +-------------------------------+
                                00ED    528 ;           |   packet AST parameter        |   :XF$L_CMT_PASTPM
                                00ED    529 ;           +-------------------------------+
                                00ED    530 ;           |               | flags | datart|   :XF$B_CMT_RATE  :XF$B_CMT_FLAGS
                                00ED    531 ;           +-------------------------------+
                                00ED    532 ;           |addr to receive addr of go bit |   :XF$L_CMT_GBITAD
                                00ED    533 ;           +-------------------------------+
                                00ED    534 ;
                                00ED    535 ;This command table is embedded in the CONTXT array
                                00ED    536 ; (offset: CTX$B_CMDTBL).  The first 4 longwords have already been
                                00ED    537 ;initialized by XF$SETUP.  Now build the remainder of the table.
                                00ED    538 ;--
                                00ED    539
                      30 A6     7C    00ED    540                CLRQ    <CTX$B_CMDTBL + XF$L_CMT_PASTAD>(R6)    ;zero AST fields
                      44 A6     7C    00F0    541                CLRQ    CTX$L_PKTAST(R6)
                      39 A6     94    00F3    542                CLRB    <CTX$B_CMDTBL + XF$B_CMT_FLAGS>(R6)     ;flags default
                42 A6    15     9B    00F6    543                MOVZBW  #EFN_DEF, CTX$W_EFN(R6) ;assume event flag # defaulted
                      3C A6     DE    00FA    544                MOVAL   <CTX$B_CMDTBL + XF$L_CMT_GBITAD>(R6),-  ;request go bit
                      3C A6           00FD    545                        <CTX$B_CMDTBL + XF$L_CMT_GBITAD>(R6)    ;addr in here
                                      00FF    546
```

```
                        00FF    548
                        00FF    549 ;++
                        00FF    550 ;Determine if an AST routine is supplied.  If so, store in the
                        00FF    551 ;command table the address of a pre- AST routine, which is part of the
                        00FF    552 ;support package.  This pre- AST routine will take the AST, and after
                        00FF    553 ;some checks call the user AST routine.  The AST parameter in the
                        00FF    554 ;command table will point to 2 longwords elsewhere in the CONTXT
                        00FF    555 ;array, which will contain the address of the user AST routine and its
                        00FF    556 ;parameter.
                        00FF    557 ;--
                        00FF    558 PKTAST_TEST:
                        00FF    559         DEFAULT_TEST    <PKTAST/4>, ASSIGN_CHN, EFN_TEST
  00000001D1'EF   DE    0109    560         MOVAL   PRE_AST,-               ;put pre- AST routine address in
        30 A6           010F    561                 <CTX$B_CMDTBL + XFSL_CMT_PASTAD>(R6) ;command table
34 A6   44 A6   DE      0111    562         MOVAL   CTX$L_PKTAST(R6), -     ;pre- AST parm is a pointer to
                        0116    563                 <CTX$B_CMDTBL + XFSL_CMT_PASTPM>(R6) ;user AST address
44 A6   0C AC   D0      0116    564         MOVL    PKTAST(AP), -           ;put user AST addr into CONTXT
                        011B    565                 CTX$L_PKTAST(R6)
                        011B    566         DEFAULT_TEST    <ASTPARM/4>, ASSIGN_CHN, EFN_TEST
48 A6   10 BC   D0      0125    567         MOVL    @ASTPARM(AP), -         ;put user AST parm in CONTXT
                        012A    568                 CTX$L_ASTPARM(R6)
                        012A    569
                        012A    570 EFN_TEST:
                        012A    571         DEFAULT_TEST    <EFN/4>, ASSIGN_CHN, MODE_TEST
42 A6   14 BC   B0      0134    572         MOVW    @EFN(AP), CTX$W_EFN(R6) ;put event flag # in CONTXT
                        0139    573
                        0139    574 MODE_TEST:
                        0139    575         DEFAULT_TEST    <MODES/4>, ASSIGN_CHN, DATART_TEST
        18 BC   90      0143    576         MOVB    @MODES(AP),-            ;put flags into command table
        39 A6           0146    577                 <CTX$B_CMDTBL + XFSB_CMT_FLAGS>(R6)
                        0148    578
                        0148    579 DATART_TEST:
                        0148    580         DEFAULT_TEST    <DATART/4>, ASSIGN_CHN, ASSIGN_CHN
                        0152    581                                         ;if < 3 args, goto ASSIGN_CHN
  38 A6   1C BC   90    0152    582         MOVB    @DATART(AP), -          ;put data rate into cmd table
                        0157    583                 <CTX$B_CMDTBL + XFSB_CMT_RATE>(R6)
    39 A6   01   88     0157    584         BISB    #XFSM_CMT_SETRTE, -     ;set data rate bit in FLAGS var
                        015B    585                 <CTX$B_CMDTBL+XFSB_CMT_FLAGS>(R6)       ;of cmd table
                        015B    586 ;++
                        015B    587 ;The command table is now complete.
                        015B    588 ;Assign a channel to the DR32.  The RMS $CREATE service with the User
                        015B    589 ;File Open option in the FOP field of the FAB is nothing more than a
                        015B    590 ;glorified assign channel, but it buys you multiple levels of logical
                        015B    591 ;name translation.
                        015B    592 ;--
                        015B    593
                        015B    594 ;initialize the FAB with the device name supplied by the caller
                        015B    595
                        015B    596 ASSIGN_CHN:
                        015B    597         DEFAULT_TEST    <DEVNAM/4>, BADPARM, BADPARM
53  00000000'EF   DE    0165    598         MOVAL   DEVICE_FAB, R3          ;R3 <- addr of FAB
    52   08 AC   D0     016C    599         MOVL    DEVNAM(AP), R2          ;R2 <- addr of devnam descriptor
```

```
                     0170   601 ;++
                     0170   602 ;The address of the FORTRAN character string descriptor is in R2.
                     0170   603 ;The descriptor look  like:
                     0170   604 ;
                     0170   605 ;        +---------------------------------+
                     0170   606 ;        |   size of char string array     |  :(R2)
                     0170   607 ;        +---------------------------------+
                     0170   608 ;        |  address of character string    |
                     0170   609 ;        +---------------------------------+
                     0170   610 ;
                     0170   611 ;If the statically declared size of the array is larger than the actual
                     0170   612 ;string, the string will be padded with blanks.  Find the true size of
                     0170   613 ;the character string before assigning the channel.
                     0170   614 ;--
  2C A3   04 A2  D0  0170   615         MOVL    4(R2), FAB$L_FNA(R3)     ;move addr of char string to FAB
2C B3   62   20  3A  0175   616         LOCC    #^O40, (R2), @FAB$L_FNA(R3) ;find first blank
    51   2C A3  C2  017A   617         SUBL2   FAB$L_FNA(R3), R1        ;R1 <- length of char string
  34 A3   51       90  017E   618         MOVB    R1, FAB$B_FNS(R3)        ;move size of string into FAB
                     01B2   619         $CREATE FAB = DEVICE_FAB         ;returns channel # in STV field
        30 50      E9  01BF   620         BLBC    R0, STAT                 ;store error status
                     0192   621 ;
                     0192   622 ;issue QIO specifying evf to be set on every packet interrupt
                     0192   623 ;
                     0192   624 10$:     $QIO_S  EFN = CTX$W_EFN(R6), -
                     0192   625                 CHAN = FAB$L_STV(R3), -
                     0192   626                 FUNC = #IO$_STARTDATA!IO$M_SETEVF, -
                     0192   627                 IOSB = CTX$Q_IOSB(R6), -         ;also embedded in CONTXT
                     0192   628                 ASTADR = @CTX$L_PRE_AST(R6), -   ;packet AST address
                     0192   629                 ASTPRM = CTX$L_PRE_PARM(R6), -
                     0192   630                 P1 = CTX$B_CMDTBL(R6), -         ;addr of command table
                     0192   631                 P2 = #XFSK_CMT_LENGTH            ;size of command table
     09 50      E9  01B6   632         BLBC    R0,STAT                  ;branch if QIO was unsuccessful
3C B6   01      90  01B9   633         MOVB    #1,@<CTX$B_CMDTBL+XFSL_CMT_GBITAD>(R6)  ;set GO bit in case
                     01BD   634                                          ;there are packets already on INPUTQ
        03      11  01BD   635         BRB     STAT                     ;R0 contains status of QIO call
                     01BF   636 ;
                     01BF   637 BADPARM:
   50   14      3C  01BF   638         MOVZWL  #SS$_BADPARAM, R0        ;needed argument defaulted
                     01C2   639 STAT:
                     01C2   640         DEFAULT_TEST    <STATUS/4>, END_STARTDEV, END_STARTDEV
20 BC   50      D0  01CC   641         MOVL    R0, @STATUS(AP)          ;store status
                     01D0   642 END_STARTDEV:
                04   01D0   643         RET
```

```
                                .SBTTL  PRE_AST -- pre - user AST routine
                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       Check if the AST routine is interrupting  critical code in the
                        ;       main routine; that is, if it could leave the list of free
                        ;       memory in an invalid state.
                        ;       If so, turn off AST's, reschedule this AST, and return.
                        ;       If not, call the user - specified AST routine.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       CALLS/G PRE_AST (ASTPARM)
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       ASTPARM points to two consecutive longwords containing
                        ;       the address of the user's AST and the user ASTPARM.
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       NONE
                        ;
                        ; OUTPUT PARAMETERS:
                        ;
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       NONE
                        ;
                        ; COMPLETION CODES:
                        ;
                        ;       NONE
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       NONE
                        ;
                        ;--
```

```
                          01D1   685 PRE_AST:
                    0000  01D1   686         .WORD   0
51    04 AC      D0     01D3   687         MOVL    4(AP), R1               ;R1 <- addr of quadword
                          01D7   688                                         ;containing addr of PKTAST
                          01D7   689                                         ;and ASTPARM
15 51    00      E4     01D7   690         BBSC    #CRITICAL_BIT, R1, -    ;determine if interrupting
                          01DB   691                 IMMEDIATE_EXIT       ;critical code; if so, exit
                          01DB   692 ;++
                          01DB   693 ;all OK; call user AST-level routine
                          01DB   694 ;--
                          01DB   695
    14 AC      DD     01DB   696         PUSHL   20(AP)                  ;saved PSL
    10 AC      DD     01DE   697         PUSHL   16(AP)                  ;saved PC
    0C AC      DD     01E1   698         PUSHL   12(AP)                  ;saved R1
    08 AC      DD     01E4   699         PUSHL   8(AP)                   ;saved R0
    04 A1      DD     01E7   700         PUSHL   4(R1)                   ;user AST-level parameter
00 B1    05      FB     01EA   701         CALLS   #5, @(R1)               ;call user AST-level routine
       17      11     01EE   702         BRB     END_PRE_AST
                          01F0   703 ;++
                          01F0   704 ;Come here if interrupted main routine during critical code.
                          01F0   705 ;Disable AST's and reschedule this AST.
                          01F0   706 ;The main level routine will re-enable AST's when it exits the
                          01F0   707 ;critical section of code.
                          01F0   708 ;--
                          01F0   709 IMMEDIATE_EXIT:
                          01F0   710         $SETAST_S       #0              ;disable AST's
                          01F9   711         $DCLAST_S       PRE_AST, R1     ;reschedule this AST
                          0207   712
                          0207   713 END_PRE_AST:
                    04    0207   714         RET
```

```
0208    716          .SBTTL  XF$PKTBLD
0208    717 ;++
0208    718 ; FUNCTIONAL DESCRIPTION:
0208    719 ;
0208    720 ;       (1) finds # of bytes needed for command packet
0208    721 ;       (2) searches freelist to find space for packet and allocates it
0208    722 ;       (3) builds command packet
0208    723 ;       (4) puts it on input queue
0208    724 ;       (5) sets 'go' bit
0208    725 ;
0208    726 ;               format of a command packet:
0208    727 ;        31                                0
0208    728 ;       +--------------------------------+
0208    729 ;       | self - relative forward link   |
0208    730 ;       +--------------------------------+
0208    731 ;       | self - relative backward link  |
0208    732 ;       +--------------------------------+
0208    733 ;       |pktctl |cmdctl |loglen |msglen  |   :(see below)
0208    734 ;       +--------------------------------+
0208    735 ;       |   byte count                   |   :XF$L_PKT_BFRSIZ
0208    736 ;       +--------------------------------+
0208    737 ;       |    virtual address of buffer   |   :XF$L_PKT_BFRADR
0208    738 ;       +--------------------------------+
0208    739 ;       |  residual memory byte count    |   :XF$L_PKT_RMBCNT
0208    740 ;       +--------------------------------+
0208    741 ;       |  residual DDI byte count       |   :XF$L_PKT_RDBCNT
0208    742 ;       +--------------------------------+
0208    743 ;       |  DR32 Status Longword (DSL)    |   :XF$L_PKT_DSL
0208    744 ;       +--------------------------------+
0208    745 ;       |  DR - device message           |   :XF$B_PKT_DEVMSG
0208    746 ;       |       //      //      //       |
0208    747 ;       +--------------------------------+
0208    748 ;       |       log area                 |
0208    749 ;       |       //      //      //       |
0208    750 ;       +--------------------------------+
0208    751 ;       |  address of ACTION routine     |
0208    752 ;       +--------------------------------+
0208    753 ;       |  address of ACTION parameter   |
0208    754 ;       +--------------------------------+
0208    755 ;
0208    756 ;       The log area and ACTION fields have no symbolic offset because
0208    757 ;       the length of the device message field is variable.  The third
0208    758 ;       longword of the command packet looks like this:
0208    759 ;
0208    760 ;        8                                0
0208    761 ;       +--------------------------------+
0208    762 ;       | length of device message       |   :XF$B_PKT_MSGLEN
0208    763 ;       +--------------------------------+
0208    764 ;       |   length of log area           |   :XF$B_PKT_LOGLEN
0208    765 ;       +--------------------------------+
0208    766 ;       | command control (function)     |   :XF$B_PKT_CMDCTL
0208    767 ;       +--------------------------------+
0208    768 ;       |  packet control byte           |   :XF$B_PKT_PKTCTL
0208    769 ;       +--------------------------------+
```

```
                     0208   771 ; CALLING SEQUENCE:
                     0208   772 ;
                     0208   773 ;       CALLS/G XF$PKTBLD (contxt, func, [index], [difsize], [devmsg],
                     0208   774 ;                          [devsiz], [logsiz], [modes], [action],
                     0208   775 ;                          [actparm], [status])
                     0208   776 ;
                     0208   777 ; INPUT PARAMETERS:
                     0208   778 ;
                     0208   779 ;offsets to AP
                     0208   780
00000004             0208   781         CONTXT = 4              ;context array
00000008             0208   782         FUNC = 8                ;a word containing a legal DR function
0000000C             0208   783         INDEX = 12              ;the index of a buffer in BARRAY
00000010             0208   784         DIFSIZE = 16            ;alternate byte count
00000014             0208   785         DEVMSG = 20             ;location of a device message
00000018             0208   786         DEVSIZ = 24             ;size of device message in bytes
0000001C             0208   787         LOGSIZ = 28             ;amt of space to reserve for log msg
00000020             0208   788         MODES = 32              ;flags and control bits to go in pkt
00000024             0208   789         ACTION = 36             ;address of an ACTION routine
00000028             0208   790         ACTPARM = 40            ;address of ACTION routine parameter
                     0208   791 ;
                     0208   792 ; OUTPUT PARAMETERS:
                     0208   793 ;
0000002C             0208   794         STATUS = 44             ;optional status returns (see below)
                     0208   795 ;
                     0208   796 ; IMPLICIT OUTPUTS:
                     0208   797 ;
                     0208   798 ;       NONE
                     0208   799 ;
                     0208   800 ; COMPLETION CODES:
                     0208   801 ;
                     0208   802 ;       (1) SS$_NORMAL          normal successful completion
                     0208   803 ;       (2) SS$_BADPARAM        input parameter error
                     0208   804 ;       (3) SS$_BADQUEUEHDR             INPUT queue interlock timeout
                     0208   805 ;       (4) SS$_INSFMEM         not enough space to build packet
                     0208   806 ;       (5) SHR$_NOCMDMEM       command memory not allocated
                     0208   807 ;
                     0208   808 ; SIDE EFFECTS:
                     0208   809 ;
                     0208   810 ;       NONE
                     0208   811 ;
                     0208   812 ;--
```

```
                                0208    814
                        OFFC    0208    815          .ENTRY  XF$PKTBLD       ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                020A    816
                                020A    817 ;a command packet is divided into distinct areas
                                020A    818 ;    1. hardware portion
                                020A    819 ;       a. fixed length
                                020A    820 ;       b. variable length
                                020A    821 ;    2. software portion
                                020A    822
                                020A    823 ;both the hardware and the software portions must be allocated command
                                020A    824 ;space; however, only the size of the hardware portion will be made
                                020A    825 ;known to the DR32 hardware
                                020A    826
                                020A    827 ;the majority of packet information is contained in the hardware-fixed
                                020A    828 ;portion of the command packet.  The hardware-variable portion has two
                                020A    829 ;optional variable-length fields -- the device message field
                                020A    830 ;and the log message field.  These fields can be from 0 - 256
                                020A    831 ;bytes; however, they must be an integer number of longwords.
                                020A    832 ;the software portion of the command packet contains the address of the
                                020A    833 ;ACTION routine(if specified) and the address of its parameter
                                020A    834 ;ACTPARM (if specified)
                                020A    835
                                020A    836 ;in this section of code:
                                020A    837 ;       R2 will accumulate the total # of bytes for the command packet
                                020A    838
                                020A    839 ;compute total size of command packet by determining lengths of
                                020A    840 ;variable-length and optional fields
                                020A    841
                    52    20    9A    020A    842          MOVZBL  #32, R2                 ;initialize R2 with # bytes in
                                020D    843                                          ;hardware-fixed potion of packet
                          58    7C    020D    844          CLRQ    R8                      ;initialize device message and
                          53    7C    020F    845          CLRQ    R3                      ;log area sizes to 0
                                0211    846
                                0211    847 ;if < 5 arguments, R2 contains total size of packet--goto BITS
                                0211    848 ;if DEVSIZ was defaulted, branch to LOGSIZE
                                0211    849          DEFAULT_TEST    <DEVSIZ/4>, BITS, LOGSIZE
                                021B    850                                          ;was size of device msg given?
                      53    18 BC    3C    021B    851          MOVZWL  @DEVSIZ(AP), R3         ;yes, round DEVSIZ up to
                                021F    852                                          ;longword boundary
                    58    53    07    C1    021F    853          ADDL3   #QUADWORD_MASK, R3, R8
                          58    07    CA    0223    854          BICL    #QUADWORD_MASK, R8
                  0100 8F     58    B1    0226    855          CMPW    R8, #256                ;is size of dev msg > 256?
                                03    1B    022B    856          BLEQU   10$                     ;no, branch around error
                                012E  31    022D    857          BRW     INVALID_ARG             ;yes, error
                          52    58    C0    0230    858 10$:     ADDL2   R8, R2                  ; add size to byte count
                                0233    859 ;R3 contains the actual size of the device message
                                0233    860 ;R8 contains the size rounded up to the next longword boundary
```

```
                     0233   862
                     0233   863 ;if < 6 arguments supplied, R2 contains total size-- goto BITS
                     0233   864 ;if LOGSIZ was defaulted, branch to see if ACTION routine was given
                     0233   865 LOGSIZE:
                     0233   866         DEFAULT_TEST    <LOGSIZ/4>, BITS, ACTION_ROUTINE
                     023D   867                                         ;was size of log message given?
                     023D   868
      54   1C BC    3C   023D   869         MOVZWL  @LOGSIZ(AP), R4     ;yes, round LOGSIZ up to
                     0241   870                                         ;longword boundary
   59 54   07 C1      0241   871         ADDL3   #QUADWORD_MASK, R4, R9
      59   07 CA      0245   872         BICL    #QUADWORD_MASK, R9
 0100 8F   59 B1      0248   873         CMPW    R9, #256                ;is size of log msg > 256 bytes?
         03 1B        024D   874         BLEQU   GO                      ;no, branch around error path
       010C 31        024F   875 BR:     BRW     INVALID_ARG             ;yes, error
      52 59 C0        0252   876 GO:     ADDL2   R9,R2                   ; no, add size to byte count
                     0255   877 ;R4 contains the actual size of the space reserved for log message
                     0255   878 ;R9 contains the size rounded up to the next longword boundary
                     0255   879
                     0255   880 ACTION_ROUTINE:
                     0255   881         DEFAULT_TEST    <ACTION/4>, BITS, BITS
                     025F   882                                         ;was ACTION routine given?
                     025F   883                                         ;if no, branch to BITS
      52   08 C0      025F   884         ADDL2   #8, R2                  ;yes,add 4 bytes for ACTION addr
                     0262   885                                         ; + 4 bytes for ACTPARM address
                     0262   886
                     0262   887
                     0262   888 ;at this point:
                     0262   889 ;       R2 contains the number of bytes needed for command packet
                     0262   890
                     0262   891 BITS:
   56   04 AC    D0   0262   892         MOVL    CONTXT(AP), R6          ;address of CONTXT array in R6
   5A   24 A6    D0   0266   893         MOVL    CTX$L_CMDBLK(R6), R10   ;R10 <- addr of command block
         03 12        026A   894         BNEQ    2$                      ;is command area allocated?
       010B 31        026C   895         BRW     TRANSFER_HALTED         ;no, return error
   51   5C A6    DE   026F   896 2$:     MOVAL   CTX$L_FREELIST(R6), R1  ;R1 <- addr of freelist head
                     0273   897
        0118 30      0273   898         BSBW    XFS$ALOCPKT             ;input: # bytes in R2
                     0276   899                                         ;       freelist head in R1
                     0276   900                                         ;output: addr of pkt in R1
      03 50 E8       0276   901         BLBS    R0, 5$                  ;test low bit for error
       00F7 31       0279   902         BRW     NO_MEM                  ;not enough space
                     027C   903
      57   51 D0      027C   904 5$:     MOVL    R1, R7                  ;preserve addr of packet from
                     027F   905                                         ;destruction by MOVC5
      5B   52 D0      027F   906         MOVL    R2, R11                 ;save size of packet
```

```
                                0282    908 ;now build the packet
                                0282    909
                                0282    910 ;first compute the addresses of and insert the variable-length fields
                                0282    911 ;       R7:  address of command packet
                                0282    912 ;       R3:  actual size of device message (in bytes)
                                0282    913 ;       R8: size of device msg, rounded up to next longword boundary
                                0282    914 ;       R4: actual size of log area (in bytes)
                                0282    915 ;       R9: size of log area, rounded up to next longword boundary
                                0282    916
                      08 A7  53 90      0282    917 NEXT:   MOVB    R3, XFSB_PKT_MSGLEN(R7) ;put size of dev msg
                                0286    918                                             ;into packet
                      09 A7  54 90      0286    919         MOVB    R4, XFSB_PKT_LOGLEN(R7) ;put in size of log area
                      0B A7  00 90      028A    920         MOVB    #MODES_DEFAULT, XFSB_PKT_PKTCTL(R7)
                                028E    921                                             ;put default MODES into packet
                                028E    922
                                028E    923 ;size of device message is in packet, now put the message itself in
                                028E    924         DEFAULT_TEST    <DEVMSG/4>, FUNC_FIELD, FUNC_FIELD
                                0298    925                                             ;if no dev msg, goto FUNC_FIELD
                                0298    926
                                0298    927 ;move device message into packet, filling with 0's to next longword
                                0298    928 ;boundary
 20 A7   58  00  14 BC  53 2C   0298    929         MOVC5   R3, @DEVMSG(AP), #0, R8, XFSB_PKT_DEVMSG(R7)
                                02A0    930
                                02A0    931 ;add the size of the fixed portion of the command packet to the sizes
                                02A0    932 ;of the device and log message fields to get the byte offset from the
                                02A0    933 ;beginning of the command packet to the ACTION routine field
                                02A0    934
                                02A0    935 FUNC_FIELD:
                                02A0    936
                   58   20 A849  9E  02A0    937         MOVAB   XFSB_PKT_DEVMSG(R8)[R9], R8 ;R8 <- offset of ACTION
                                02A5    938
                                02A5    939 ;insert fixed-length arguments in the order they were supplied
                                02A5    940
                                02A5    941         DEFAULT_TEST    <FUNC/4>, INV, INV
                                02AF    942                                             ;if FUNC defaulted, goto
                                02AF    943                                             ;INVALID_ARG
                      0F  08 BC  B1 02AF    944         CMPW    @FUNC(AP), #15          ;function codes are from 0:15
                         03  18 02B3    945         BLEQU   OK                      ;branch around error path
                        00A6  31 02B5    946 INV:    BRW     INVALID_ARG             ;invalid function code
                   0A A7  08 BC  90 02B8    947 OK:     MOVB    @FUNC(AP), XFSB_PKT_CMDCTL(R7)
                                02BD    948                                             ;insert function code
                                02BD    949                                             ;high bits must be zero
```

```
                                02BD   951 INDEX_FIELD:
                0C A7   7C      02BD   952         CLRQ    XF$L_PKT_BFRSIZ(R7)     ;clear byte count & buffer addr
                                02C0   953                                         ;(assume no data transfer)
                                02C0   954         DEFAULT_TEST    <INDEX/4>,FIELDS_DONE,ACTION_FIELD
                                02CA   955                                         ;if < 3 args goto FIELDS_DONE
                                02CA   956                                         ;else if defaulted go to ACTION_FIELD
           51   0C BC   3C      02CA   957         MOVZWL  @INDEX(AP), R1          ;R1 <- index of buffer
                   E5   13      02CE   958         BEQL    INV                     ;index of 0 is invalid
           40 A6    51   B1     02D0   959         CMPW    R1, CTX$W_NUMBUF(R6)    ;index > number of buffers?
                    DF   1A     02D4   960         BGTRU   INV                     ;yes, invalid buffer index
                    51   D7     02D6   961         DECL    R1                      ;R1 <- buffer offset from base
                                02D8   962                                         ;of buffer array
           51   4C A6   C4      02D8   963         MULL2   CTX$L_BUFSIZ(R6), R1    ;R1 <- byte offset from base of
                                02DC   964                                         ;buffer array of this buffer
      10 A7   51   2C A6   C1   02DC   965         ADDL3   CTX$L_DATABLK(R6), R1,-
                                02E2   966                 XF$L_PKT_BFRADR(R7)     ;put buffer addr into packet
                                02E2   967         DEFAULT_TEST    <DIFSIZE/4>, 10$, 10$
         0C A7   10 BC   D0     02EC   968         MOVL    @DIFSIZE(AP), XF$L_PKT_BFRSIZ(R7)
                                02F1   969                                         ;alternate transfer byte count
                    05   11     02F1   970         BRB     ACTION_FIELD
         0C A7   4C A6   D0     02F3   971 10$:     MOVL    CTX$L_BUFSIZ(R6), XF$L_PKT_BFRSIZ(R7)
                                02F8   972                                         ;standard transfer byte count
                                02F8   973
                                02F8   974 ACTION_FIELD:
                                02F8   975         DEFAULT_TEST    <ACTION/4>,MODES_FIELD,MODES_FIELD
         0B A7   04   88        0302   976         BISB2   #XF$M_PKT_ACTBIT, -     ;set "ACTION routine given" bit
                                0306   977                 XF$B_PKT_PKTCTL(R7)     ;in packet control field
                                0306   978
                                0306   979 ;R8 contains byte offset from beginning of command packet to ACTION routine
                                0306   980 ;field of packet
                                0306   981
           58   57   C0        0306   982         ADDL2   R7,R8                   ;R8 <- addr of ACTION field
      88   24 AC   D0           0309   983         MOVL    ACTION(AP),(R8)+        ;put addr of ACTION routine into packet
                                030D   984         DEFAULT_TEST    <ACTPARM/4>,MODES_FIELD,MODES_FIELD
                                0317   985                                         ;If ACTPARM defaulted goto MODES_FIELD
      68   28 AC   D0           0317   986         MOVL    ACTPARM(AP),(R8)        ;put addr of ACTPARM in packet
                                031B   987
                                031B   988 MODES_FIELD:
                                031B   989         DEFAULT_TEST    <MODES/4>,FIELDS_DONE,FIELDS_DONE
                                0325   990                                         ;if MODES is defaulted, goto
                                0325   991                                         ;       FIELDS_DONE
         0B A7   00   8A        0325   992         BICB2   #MODES_DEFAULT, XF$B_PKT_PKTCTL(R7)
                                0329   993                                         ;clear out default modes settings but
                                0329   994                                         ;preserve "action routine present" bit
   0B A7   20 BC   88           0329   995         BISB2   @MODES(AP), XF$B_PKT_PKTCTL(R7)
                                032E   996                                         ;sets   (1) interrupt control
                                032E   997                                         ;       (2)length error bit
                                032E   998                                         ;       (3) pkt control bits
                                032E   999                                         ;to user-supplied values
                                032E  1000
                                032E  1001 ;the packet is now completely built and ready to be put on the input queue
                                032E  1002
      20 BC   08   E1           032E  1003         BBC     #XF$V_PKT_HT, @MODES(AP),-
                   11           0332  1004                 INSERT_AT_TAIL          ;clear bit <==> tail
                                0333  1005 INSERT_AT_HEAD:
                                0333  1006 ;R10 contains the address of the command block
                                0333  1007
```

```
                     0333  1008          QRETRY -
                     0333  1009          INSQHI  (R7), CMD$L_INPTQ(R10) -;attempt insertion
                     0333  1010                  SUCCESS = SET_GO_BIT -
                     0333  1011                  ERROR = Q_FAILURE       ;exceeded retry limit
                     0344  1012
                     0344  1013 FIELDS_DONE:
                     0344  1014 INSERT_AT_TAIL:
                     0344  1015          QRETRY -
                     0344  1016          INSQTI  (R7), CMD$L_INPTQ(R10) -;attempt insertion at tail
                     0344  1017                  ERROR = Q_FAILURE       ;exceeded retry limit
                     0355  1018
                     0355  1019 SET_GO_BIT:
                     0355  1020
   3C B6   01   90   0355  1021          MOVB    #1, @<CTX$B_CMDTBL+XFSL_CMT_GBITAD>(R6)
                     0359  1022                                          ;notify the Dr that there is a
                     0359  1023                                          ;packet on the INPUT queue
                     0359  1024
      50   01   3C   0359  1025          MOVZWL  #SS$_NORMAL, R0         ;success status return
           21   11   035C  1026          BRB     STORE_STAT              ;branch around error paths
                     035E  1027 INVALID_ARG:
      50   14   3C   035E  1028          MOVZWL  #SS$_BADPARAM, R0       ;input parameter error
           05   11   0361  1029          BRB     DEALLOCATE
                     0363  1030 Q_FAILURE:
   50   0394 8F   3C   0363  1031          MOVZWL  #SS$_BADQUEUEHDR, R0    ;interlocked queue timeout
                     0368  1032 DEALLOCATE:                               ;inputs to XFSSDEALOCPKT:
   51   5C A6   DE   0368  1033          MOVAL   CTX$L_FREELIST(R6), R1  ;R1: address of freelist head
      53   58   D0   036C  1034          MOVL    R11, R3                 ;R3: size of packet in bytes
                     036F  1035                                          ;R7: address of packet
           61   10   036F  1036          BSBB    XFSSDEALOCPKT           ;deallocate the packet
           0C   11   0371  1037          BRB     STORE_STAT
                     0373  1038 NO_MEM:
   50   0124 8F   3C   0373  1039          MOVZWL  #SS$_INSFMEM, R0        ;not enough space to build pkt
           05   11   0378  1040          BRB     STORE_STAT
                     037A  1041 TRANSFER_HALTED:
   50   1278 8F   3C   037A  1042          MOVZWL  #SHR$_NOCMDMEM, R0      ;command memory not allocated
                     037F  1043 STORE_STAT:
                     037F  1044          DEFAULT_TEST    <STATUS/4>, END_PKTBLD, END_PKTBLD
                     0389  1045                                          ;was STATUS arg given?
     2C BC   50   D0   0389  1046          MOVL    R0, @STATUS(AP) ;yes, store status return
                     038D  1047 END_PKTBLD:
           04   038D  1048          RET
```

```
                        .SBTTL  XFSSALOCPKT -- ALLOCATE A COMMAND PACKET
                                                AND RETURN ITS ADDRESS
                ;
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is called by XFSPKTBLD to allocate a command
                ;       packet.  It searches the list of free chunks of command
                ;       space to find the required amount of memory.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    XFSSALOCPKT
                ;
                ; INPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       R1 contains the address of a pointer to the free list
                ;       R2 contains the number of bytes needed for packet
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       R1 contains the address of the allocated packet
                ;
                ; COMPLETION CODES:
                ;
                ;       returned in R0 : not enough memory available
                ;                        1 = sucess
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ;--
```

```
                     038E  1091 XFS$ALOCPKT::                           ;allocate memory
                0C   BB  038E  1092         PUSHR   #^M<R2,R3>
                     0390  1093
                     0390  1094 ;Since command packets must be quadword aligned, the allocation
                     0390  1095 ;granularity of each packet is 8 bytes.
                     0390  1096
              52     07  C0  0390  1097         ADDL2   #GRANULARITY, R2        ;round size up to next
              52     07  CA  0393  1098         BICL2   #GRANULARITY, R2        ;quadword boundary
              50     51  D0  0396  1099         MOVL    R1, R0                  ;copy address of first free
                     0399  1100                                                 ;block address
           34 A6     01  88  0399  1101         BISB2   #CRITICAL_MASK, -       ;set bit in AST parm to indicate
                     039D  1102                 <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6)
                     039D  1103                                                 ;"entering critical code"
                     039D  1104 ;Find a piece of memory large enough for requested allocation.
                     039D  1105
              51     50  D0  039D  1106 10$:     MOVL    R0, R1                  ;save addr of previous free blk
              50     61  D0  03A0  1107         MOVL    (R1), R0                ;get addr of next free block
                     2A  13  03A3  1108         BEQL    END_ALOCPKT             ;if equal no memory available
           04 A0     52  D1  03A5  1109         CMPL    R2, 4(R0)               ;free block big enough?
                     F2  1A  03A9  1110         BGTRU   10$                     ;no, go try next block
                     03AB  1111
                     03AB  1112 ;free block found
                     03AB  1113
                     0E  13  03AB  1114         BEQL    EQUAL                   ;if eql free block is exact size
                     03AD  1115
                     03AD  1116 ;Free block is bigger than requested allocation.  Allocate what was
                     03AD  1117 ;asked for and put remainder of block back on free list.
                     03AD  1118
           53  52  50  C1  03AD  1119         ADDL3   R0, R2, R3              ;R3 <- addr of new free block
                     03B1  1120
              83     80  D0  03B1  1121         MOVL    (R0)+, (R3)+            ;copy link to new free block
           63 60     52  C3  03B4  1122         SUBL3   R2, (R0), (R3)          ;calc size of new free block
              70     73  DE  03B8  1123         MOVAL   -(R3), -(R0)            ;set link to new free block
                     03BB  1124
                     03BB  1125 ;Remove block from free list.
                     03BB  1126 EQUAL:
              61     60  D0  03BB  1127         MOVL    (R0), (R1)              ;copy link to new free block
              51     80  9E  03BE  1128         MOVAB   (R0)+, R1               ;R1 <- addr of allocated blk
                     03C1  1129                                                 ;R0 indicates success
        09 34 A6     00  E4  03C1  1130         BBSC    #CRITICAL_BIT, -        ;did AST interrupt critical code
                     03C6  1131                 <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6), -
                     03C6  1132                 END_ALOCPKT                     ;if not, branch to END_ALOCPKT
                     03C6  1133         $SETAST_S       #1                      ;if so, the AST routine disabled
                     03CF  1134                                                 ;AST's and rescheduled itself,so
                     03CF  1135                                                 ;upon exiting critical code,
                     03CF  1136                                                 ;re-enable AST's
                     03CF  1137 END_ALOCPKT:
                0C   BA  03CF  1138         POPR    #^M<R2,R3>
                     05  03D1  1139         RSB
```

```
                        .SBTTL  XF$$DEALOCPKT -- DEALLOCATE COMMAND PACKET
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is called by XF$GETPKT and XF$PKTBLD to return
                ;       the memory used for a command packet.  It searches the list
                ;       of free blocks of memory to find where to return the packet
                ;       memory, and agglomerates the returned memory with adjacent
                ;       blocks if possible.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       NONE
                ;
                ; INPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       R1 = address of allocation region listhead
                ;       R3 = size of blocks in bytes
                ;       R7 = address of block to be deallocated
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;
                ;       R1, R3, and R7 are destroyed
                ;
                ;--
```

```
                         03D2  1183 XFS$DEALOCPKT::
            52   DD      03D2  1184         PUSHL   R2
       53   07   C0      03D4  1185         ADDL2   #GRANULARITY, R3          ;round size up to next
       53   07   CA      03D7  1186         BICL2   #GRANULARITY, R3          ;quadword boundary
 34    A6   01   88      03DA  1187         BISB2   #CRITICAL_MASK, -          ;set bit in AST parm to indicate
                         03DE  1188                 <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6)
                         03DE  1189                                           ;"entering critical code"
                         03DE  1190
                         03DE  1191 ;Find where in free list to return the memory.
                         03DE  1192
            52   51      D0    03DE  1193 10$:    MOVL    R1, R2                ;R2 <- addr of prev free block
            51   62      D0    03E1  1194         MOVL    (R2), R1              ;R1 <- addr of next free block
                 05      13    03E4  1195         BEQL    20$                   ;if equal, end of list
            51   57      D1    03E6  1196         CMPL    R7, R1                ;block logically go here?
                 F3      1A    03E9  1197         BGTRU   10$                   ;no, keep looking
                         03EB  1198
                         03EB  1199 ;Determine of returned memory can be agglomerated with the block of
                         03EB  1200 ;memory immediately following it.
                         03EB  1201
            67   51      D0    03EB  1202 20$:    MOVL    R1, (R7)              ;assume no agglomeration
       7E   53   57      C1    03EE  1203         ADDL3   R7, R3, -(SP)         ;calculate addr of end of block
            8E   51      D1    03F2  1204         CMPL    R1, (SP)+             ;end of block = next in list?
                 06      12    03F5  1205         BNEQ    30$                   ;if neq do not agglomerate
            67   81      D0    03F7  1206         MOVL    (R1)+, (R7)           ;move link to block being freed
            53   61      C0    03FA  1207         ADDL2   (R1), R3              ;R3 <- length of new free block
                         03FD  1208
                         03FD  1209 ;Determine if returned memory can be agglomerated with the block of
                         03FD  1210 ;memory immediately preceeding it.
                         03FD  1211
            52   DD      03FD  1212 30$:    PUSHL   R2                    ;calc end addr of previos block
       82   57   D0      03FF  1213         MOVL    R7, (R2)+             ;assume no agglomeration
       6E   62   C0      0402  1214         ADDL2   (R2), (SP)            ;add length to block base addr
       8E   57   D1      0405  1215         CMPL    R7, (SP)+             ;end addr = block being freed?
            09   12      0408  1216         BNEQ    40$                   ;no, do not agglomerate blocks
       53   62   C0      040A  1217         ADDL2   (R2), R3              ;R3 <- size of new free block
       72   67   D0      040D  1218         MOVL    (R7), -(R2)           ;move link to previous free blk
       57   52   D0      0410  1219         MOVL    R2, R7                ;set addr of new free block
                         0413  1220
 04    A7   53   D0      0413  1221 40$:    MOVL    R3, 4(R7)             ;set size of free block
09 34    A6   00   E4      0417  1222         BBSC    #CRITICAL_BIT, -      ;did AST interrupt critical code
                         041C  1223                 <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6), -
                         041C  1224                 END_DEALOCPKT
                         041C  1225         $SETAST_S       #1                ;if so, the AST routine disabled
                         0425  1226                                           ;AST's and rescheduled itself,so
                         0425  1227                                           ;upon exiting critical code,
                         0425  1228                                           ;re-enable AST's
                         0425  1229
                         0425  1230 END_DEALOCPKT:
            04   BA      0425  1231         POPR    #^M<R2>
                 05      0427  1232         RSB
```

```
                     0428  1234          .SBTTL  XFSFREESET -- PUT PACKETS ON FREEQ
                     0428  1235 ;++
                     0428  1236 ; FUNCTIONAL DESCRIPTION:
                     0428  1237 ;
                     0428  1238 ;       Determine the size of the packets to be released onto the FREE
                     0428  1239 ;       queue according to input arguments.  Then build the number of
                     0428  1240 ;       packets specified and release them onto the FREE queue.
                     0428  1241 ;
                     0428  1242 ; CALLING SEQUENCE:
                     0428  1243 ;
                     0428  1244 ;       CALLS/G  XFSFREESET(contxt, [numpkt], [intctrl], [action], -
                     0428  1245 ;                                   [actparm], [status])
                     0428  1246 ;
                     0428  1247 ; INPUT PARAMETERS:
                     0428  1248 ;
                     0428  1249 ;offsets to AP:
00000004             0428  1250          CONTXT = 4              ;context array
00000008             0428  1251          NUMPKT = 8              ;number of packets to put on FREEQ
0000000C             0428  1252          INTCTRL = 12            ;interrupt control bits to put in pkt
00000010             0428  1253          ACTION = 16             ;address of ACTION routine
00000014             0428  1254          ACTPARM = 20            ;address of ACTION parameter
                     0428  1255 ;
                     0428  1256 ; IMPLICIT INPUTS:
                     0428  1257 ;
                     0428  1258 ;       NONE
                     0428  1259 ;
                     0428  1260 ; OUTPUT PARAMETERS:
                     0428  1261 ;
                     0428  1262 ;offsets to AP:
00000018             0428  1263          STATUS = 24             ;status returns (see completion codes)
                     0428  1264 ;
                     0428  1265 ; IMPLICIT OUTPUTS:
                     0428  1266 ;
                     0428  1267 ;       NONE
                     0428  1268 ;
                     0428  1269 ; COMPLETION CODES:
                     0428  1270 ;
                     0428  1271 ;       (1) SS$_NORMAL          normal successful completion
                     0428  1272 ;       (2) SS$_BADQUEUEHDR             INPUT queue interlock timeout
                     0428  1273 ;       (3) SS$_INSFMEM         not enough memory to build packet
                     0428  1274 ;       (4) SHR$_NOCMDMEM       command memory is not allocated
                     0428  1275 ;
                     0428  1276 ; SIDE EFFECTS:
                     0428  1277 ;
                     0428  1278 ;       NONE
                     0428  1279 ;
                     0428  1280 ;--
```

```
                07FC  0428  1282          .ENTRY  XF$FREESET      ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
56   04 AC       D0  042A  1283          MOVL    CONTXT(AP), R6          ;R6 <- addr of CONTXT
5A   24 A6       D0  042E  1284          MOVL    CTX$L_CMDBLK(R6), R10   ;R10 <- addr of command area
          03     12  0432  1285          BNEQ    FIND_SIZE               ;if addr of command area = 0,
                     0434  1286                                          ;transfer is halted
          00AD   31  0434  1287          BRW     TRANS_HALTED            ;error path
                     0437  1288
                     0437  1289
                     0437  1290
                     0437  1291 ;++
                     0437  1292 ;determine the size of the packets to be released onto the FREEQ by
                     0437  1293 ;looking at the input arguments
                     0437  1294 ;--
                     0437  1295
                     0437  1296 ;find size of field to reserve for device message
                     0437  1297
                     0437  1298 FIND_SIZE:
52   5A A6          3C  0437  1299          MOVZWL  CTX$W_IDEVSIZ(R6), R2   ;R2 <- size of dev msg
   52   07          C0  043B  1300          ADDL2   #QUADWORD_MASK, R2      ;round up to quadword boundary
   52   07          CA  043E  1301          BICL    #QUADWORD_MASK, R2      ;R2 <- size of devmsg field
   52   20          C0  0441  1302          ADDL2   #32, R2                 ;R2 <- size of command packet
                     0444  1303
                     0444  1304 ;determine if ACTION routine and ACTPARM are to be put in command pkt
                     0444  1305
   57   52          D0  0444  1306          MOVL    R2, R7                  ;R7 <- offset of ACTION routine
        53          7C  0447  1307          CLRQ    R3                      ;assume no ACTION or ACTPARM
                     0449  1308          DEFAULT_TEST    <ACTION/4>, 5$, 5$
                     0453  1309                                          ;if defaulted, goto 5$
53   10 AC          D0  0453  1310          MOVL    ACTION(AP), R3          ;R3 <- addr of ACTION routine
   52   08          C0  0457  1311          ADDL2   #8, R2                  ;add sizes of ACTION and ACTPARM
                     045A  1312                                          ;to total packet size
                     045A  1313          DEFAULT_TEST    <ACTPARM/4>, 5$, 5$
                     0464  1314                                          ;if defaulted, goto 5$
54   14 AC          D0  0464  1315          MOVL    ACTPARM(AP), R4 ;R4 <- addr of ACTPARM
                     0468  1316
                     0468  1317 ;find the interrupt control bits to be put in packet
                     0468  1318
   58   00          9A  0468  1319 5$:      MOVZBL  #INT_DEFAULT, R8        ;default interrupt ctrl setting
                     0468  1320          DEFAULT_TEST    <INTCTRL/4>, 10$, 10$
                     0475  1321                                          ;if defaulted goto 10$
58   0C BC          90  0475  1322          MOVB    @INTCTRL(AP), R8        ;R8 <- interrupt control bits
                     0479  1323
                     0479  1324 ;find the number of packets to be put onto the FREEQ
                     0479  1325
   55   01          9A  0479  1326 10$:     MOVZBL  #1, R5                  ;default # of pkts to put on
                     047C  1327                                          ;FREEQ
                     047C  1328          DEFAULT_TEST    <NUMPKT/4>, ANOTHER_PKT, ANOTHER_PKT
                     0486  1329                                          ;if defaulted goto ANOTHER_PKT
55   08 BC          3C  0486  1330          MOVZWL  @NUMPKT(AP), R5         ;R5 <- # of pkts to put on queue
```

```
                                048A  1332 ;++
                                048A  1333 ;build a packet
                                048A  1334 ;--
                                048A  1335 ANOTHER_PKT:
                 51   5C A6    DE  048A  1336         MOVAL   CTX$L_FREELIST(R6), R1  ;R1 <- addr of ptr to freelist
                        FEFD    30  048E  1337         BSBW    XF$$ALOCPKT             ;input: size of pkt in R2
                                    0491  1338                                         ;       freelist ptr in R1
                                    0491  1339                                         ;returns addr of pkt in R1
                      49 50     E9  0491  1340         BLBC    R0, NOT_MEM             ;not enough memory to build in
                         3E     BB  0494  1341         PUSHR   #^M<R1,R2,R3,R4,R5>     ;preserve from MOVC5
    52   00   00000200'EF       00  0496  1342         MOVC5   #0, DUMMY_ADR, #0, R2, (R1)     ;zero the packet
                                61  049F
                         3E     BA  04A0  1343         POPR    #^M<R1,R2,R3,R4,R5>     ;restore registers
                                    04A2  1344
                                    04A2  1345 ;put address of ACTION routine and addr of ACTPARM into packet
                                    04A2  1346 ;R3 contains addr of ACTION routine, R4 contains addr of parameter
                                    04A2  1347 ;R7 contains offset from beginning of packet to ACTION routine field
                                    04A2  1348
                         53     D5  04A2  1349         TSTL    R3                      ;addr of ACTION routine
                         0B     13  04A4  1350         BEQL    5$                      ;no ACTION routine if addr = 0
                    59   6147   9E  04A6  1351         MOVAB   (R1)[R7], R9            ; R9 <- addr of ACTION field
                      69   53     7D  04AA  1352         MOVQ    R3, (R9)                ;put ACTION and ACTPARM in pkt
                 0B A1     04     88  04AD  1353         BISB2   #XF$M_PKT_ACTBIT, -     ;set "ACTION routine given" bit
                                    04B1  1354                 XF$B_PKT_PKTCTL(R1)     ;in packet control field of pkt
                                    04B1  1355
                                    04B1  1356 ;insert interrupt control bits into packet
                                    04B1  1357
                    06   58     F0  04B1  1358 5$:     INSV    R8, #XF$V_PKT_INTCTL,-  ;put interrupt control bits
                 0B A1     02         04B4  1359                 #XF$S_PKT_INTCTL, XF$B_PKT_PKTCTL(R1)   ;into packet
                                    04B7  1360
                                    04B7  1361 ;put size of device message into packet
                                    04B7  1362
    08 A1     5A A6     90          04B7  1363         MOVB    CTX$W_IDEVSIZ(R6), XF$B_PKT_MSGLEN(R1)
                                    04BC  1364
                                    04BC  1365 ;put packet onto FREEQ
                                    04BC  1366
                                    04BC  1367         QRETRY -
                                    04BC  1368         INSQTI  (R1), CMD$L_FREEQ(R10) -;attempt to insert packet
                                    04BC  1369                 ERROR = BAD_QUEUE       ;exceeded retry limit
                                    04CE  1370
                                    04CE  1371 A_OK:                                   ;packet is on queue
                                    04CE  1372
                      B9 55     F5  04CE  1373         SOBGTR  R5, ANOTHER_PKT         ;go do another packet
                                    04D1  1374
                                    04D1  1375
```

```
                                04D1  1377 ;++
                                04D1  1378 ;all the packets have been successfully inserted onto the FREEQ
                                04D1  1379 ;--
                                04D1  1380
          50    01    3C      04D1  1381          MOVZWL  #SS$_NORMAL, R0         ;success status return
                13    11      04D4  1382          BRB     END_FREESET
                              04D6  1383
                              04D6  1384 BAD_QUEUE:
     50   0394  8F    3C      04D6  1385          MOVZWL  #SS$_BADQUEUEHDR, R0    ;interlock timeout
                0C    11      04DB  1386          BRB     END_FREESET
                              04DD  1387 NOT_MEM:
     50   0124  8F    3C      04DD  1388          MOVZWL  #SS$_INSFMEM, R0        ;not enough command space
                05    11      04E2  1389          BRB     END_FREESET
                              04E4  1390 TRANS_HALTED:
     50   1278  8F    3C      04E4  1391          MOVZWL  #SHR$_NOCMDMEM, R0      ;transfer halted; command
                              04E9  1392                                          ;space deallocated
                              04E9  1393 END_FREESET:
                              04E9  1394          DEFAULT_TEST    <STATUS/4>, 10$, 10$
       18 BC    50    D0      04F3  1395          MOVL    R0, @STATUS(AP)         ;store status
                04    04F7  1396 10$:    RET
```

```
                                .SBTTL  XF$GETPKT -- GET A PACKET
                                        FROM THE TERMINATION QUEUE
                ;
                ;++
                ; FUNCIONAL DESCRIPTION:
                ;
                ;       Attempt to remove a packet from the TERMQ. If successful, break
                ;the packet up into its various fields and return them to the caller. If
                ;an ACTION routine is specified in the packet, call it.  Finally, return
                ;the memory that was used to build this packet.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLS/G XF$GETPKT (contxt, [waitflg], [func], [index], -
                ;                          [devflag], [logflag], [status])
                ;
                ; INPUT PARAMETERS:
                ;
                ;offsets to AP:
00000004        CONTXT = 4                      ;context array
00000008        WAITFLG = 8                     ;wait for event flag/immediate return
                ;
                ; IMPLICIT INPUTS:
                ;
                ;fields in the CONTXT array:
                ;       CTX$L_DATABLK
                ;       CTX$W_NUMBUF
                ;       CTX$L_IDEVMSG
                ;       CTX$L_IDEVSIZ
                ;       CTX$L_ILOGMSG
                ;       CTX$L_ILOGSIZ
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;offsets to AP:
0000000C        FUNC = 12                       ;function specified in packet
00000010        INDEX = 16                      ;buffer index specified in packet
00000014        DEVFLAG = 20                    ;set if device message in packet
00000018        LOGFLAG = 24                    ;set if log message in packet
0000001C        STATUS = 28                     ;status return
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;fields in the CONTXT array:
                ;       CTX$L_MEMCNT
                ;       CTX$L_DDICNT
                ;       CTX$L_DSL
                ;
```

```
                        ; COMPLETION CODES:
                        ;
                        ;       (1) SS$_NORMAL          normal successful completion
                        ;       (2) SS$_BADQUEUEHDR             TERM queue interlocked timeout
                        ;       (3) SHR$_HALTED         XFSCLEANUP was called
                        ;       (4) SHR$_QEMPTY(=0)     no packet, but transfer still going
                        ;       (5) SHR$_NOCMDMEM       no command memory was allocated at the
                        ;                               time of the call to this routine
                        ;       (6) status of ACTION routine
                        ;               XFSGETPKT's status is an input to the ACTION routine.
                        ;               The ACTION routine may overwrite the status argument
                        ;               with a status return of its own.
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       If XFSCLEANUP was called, neither the command packets nor the
                        ;       queues are any longer accessible.
                        ;
                        ;--
```

```
                                04F8  1466
                          0FFC  04F8  1467          .ENTRY  XF$GETPKT       ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                04FA  1468
                     56   04 AC   D0  04FA  1469          MOVL    CONTXT(AP), R6          ;R6 <- addr of CONTXT
                                04FE  1470
                          52    D4  04FE  1471          CLRL    R2                      ;assume WAITFLG defaulted
                                0500  1472          DEFAULT_TEST    <WAITFLG/4>, 10$, 10$
                                050A  1473                                              ;default is event flag wait
                     52   08 BC   9A  050A  1474          MOVZBL  @WAITFLG(AP), R2        ;input to GET_ADDR
                           00EA   30  050E  1475 10$:     BSBW    GET_ADDR                ;returns addr of pkt in R7, if
                                0511  1476                                              ;there is  a pkt on TERMQ
                          03 50   E8  0511  1477          BLBS    R0, DISSECT_PKT         ;status returned in R0
                           00A1   31  0514  1478          BRW     STORE_STATUS            ;error in removing pkt
                                0517  1479                                              ;from TERMQ
                                0517  1480
                                0517  1481
                                0517  1482 ;++
                                0517  1483 ;come here if successfully removed a packet from TERMQ
                                0517  1484 ;
                                0517  1485 ;give the user as many command packet fields as he supplied variables
                                0517  1486 ;to hold
                                0517  1487 ;--
                                0517  1488 DISSECT_PKT:
                                0517  1489
                                0517  1490 ;++
                                0517  1491 ;compute sizes of device and log message fields
                                0517  1492 ;
                                0517  1493 ;the "actual size" of the device message is the number of bytes
                                0517  1494 ;specified by the DEVSIZ argument in the call to XF$PKTBLD.  This is the
                                0517  1495 ;value stored in the XF$B_PKT_MSGLEN field of the packet.
                                0517  1496 ;the "packet size" of the device message refers to the fact that the
                                0517  1497 ;device message field is 0-filled to the next longword boundary in the
                                0517  1498 ;packet.  A similar situation occurs with the log message field.
                                0517  1499 ;--
                                0517  1500
                  58    08 A7   9A  0517  1501          MOVZBL  XF$B_PKT_MSGLEN(R7), R8 ;R8 <- actual size of device msg
                59  58    07   C1  051B  1502          ADDL3   #QUADWORD_MASK, R8, R9  ;round size up to longword bound
                      59    07   CA  051F  1503          BICL    #QUADWORD_MASK, R9      ;R9 <- packet size of dev msg
                                0522  1504
                  5A    09 A7   9A  0522  1505          MOVZBL  XF$B_PKT_LOGLEN(R7), R10 ;R10 <- actual size of log msg
                5B  5A    07   C1  0526  1506          ADDL3   #QUADWORD_MASK, R10, R11 ;round up to longword boundary
                      5B    07   CA  052A  1507          BICL    #QUADWORD_MASK, R11     ;R11 <- packet size of log msg
                                052D  1508
```

```
                                                G 6
                     052D  1510          DEFAULT_TEST    <FUNC/4>, TRANSFER_STATUS, INDEX_TEST
                     0537  1511                                  ;if < 3 args goto TRANSFER_STATUS
                     0537  1512                                  ;else if FUNC defaulted goto INDEX_TEST
                     0537  1513
                     0537  1514 ;store function from packet into supplied argument
                     0537  1515
 0C BC   0A A7   9B  0537  1516          MOVZBW  XF$B_PKT_CMDCTL(R7), @FUNC(AP)
                     053C  1517
                     053C  1518
                     053C  1519
                     053C  1520 INDEX_TEST:
                     053C  1521          DEFAULT_TEST    <INDEX/4>, TRANSFER_STATUS, DEVFLAG_TEST
                     0546  1522                                  ;if < 4 args goto TRANSFER_STATUS
                     0546  1523                                  ;else if INDEX defaulted goto DEVFLAG
                     0546  1524
                     0546  1525 ;convert buffer address in packet to index
                     0546  1526
     53   10 A7   D0  0546  1527          MOVL    XF$L_PKT_BFRADR(R7),R3  ; was a data buffer transferred?
          08   13     054A  1528          BEQL    10$                     ; if addr = 0, no
53   53   2C A6   C3  054C  1529          SUBL3   CTX$L_DATABLK(R6),R3,R3 ; yes, R3 <- byte offset from base
                     0551  1530                                          ; of buffer array
     53   4C A6   C6  0551  1531          DIVL2   CTX$L_BUFSIZ(R6), R3    ;R3 <- index offset from base
          53   D6     0555  1532          INCL    R3                      ;R3 <- index of buffer
 10 BC    53   B0  0557  1533 10$:    MOVW    R3, @INDEX(AP)          ;store index
                     055B  1534
                     055B  1535
                     055B  1536
                     055B  1537 DEVFLAG_TEST:
                     055B  1538
                     055B  1539 ;determine if there is a device message in this packet
                     055B  1540 ;R8 contains actual size of device message
                     055B  1541 ;The setting of DEVFLAG is a bit convoluted; it stems from the fact
                     055B  1542 ;that there are no spare registers left to hold DEVFLAG'S future value
                     055B  1543 ;and relies on the fact that MOVC5 clears R2.
                     055B  1544
     52   FF 8F   90  055B  1545          MOVB    #TRUE, R2               ;R2 is the complement of DEVFLAG
                     055F  1546                                          ; (assume no device message)
0E 1C A7   03   E1   055F  1547          BBC     #XF$V_PKT_FREQPK, -     ;was this packet taken from the
                     0564  1548                  XF$L_PKT_DSL(R7), -     ;FREEQ (does it contain
                     0564  1549                  10$                     ;unsolicited input)?
                     0564  1550                                          ;if not, goto 10$
          50 A6   D5  0564  1551          TSTL    CTX$L_IDEVMSG(R6)       ;was the array to store the
                     0567  1552                                          ;device message given?
          09   13     0567  1553          BEQL    10$                     ;no, goto 10$
                     0569  1554
                     0569  1555 ;move the device message field from the packet into the array IDEVMSG,
                     0569  1556 ;which was specified in the call to XF$SETUP
                     0569  1557
00   20 A7   58   2C  0569  1558          MOVC5   R8, XF$B_PKT_DEVMSG(R7), #0,-
  50 B6    5A A6     056E  1559                  CTX$W_IDEVSIZ(R6), @CTX$L_IDEVMSG(R6)
                     0572  1560 10$:    DEFAULT_TEST    <DEVFLAG/4>, TRANSFER_STATUS, LOGFLAG_TEST
                     057C  1561                                  ;if < 5 args goto TRANSFER_STATUS
                     057C  1562                                  ;else if DEVFLAG defaulted goto LOGFLAG_
     14 BC   52   92 057C  1563          MCOMB   R2, @DEVFLAG(AP)        ;set DEVFLAG appropriately
```

```
                                0580  1565
                                0580  1566 LOGFLAG_TEST:
                                0580  1567
                                0580  1568 ;determine if there is a log message in this packet
                                0580  1569 ;R10 contains actual size of log message
                                0580  1570 ;the same note on the setting of DEVFLAG applies to LOGFLAG
                                0580  1571
                     52   FF 8F   90  0580  1572         MOVB    #TRUE, R2               ;R2 is the complement of LOGFLAG
                                0584  1573                                               ; (assume no log message)
                  0F 1C A7   16   E1  0584  1574         BBC     #<XF$V_PKT_DDISTS+XF$V_PKT_LOG>, -      ;is "log msg in"
                                0589  1575                 XF$L_PKT_DSL(R7), -         ;bit set in the packet?
                                0589  1576                 10$                         ;branch if not
                        54 A6   D5  0589  1577         TSTL    CTX$L_ILOGMSG(R6)       ;was the array to store the
                                058C  1578                                             ;device message given?
                           0A   13  058C  1579         BEQL    10$                     ;no, goto 10$
                                058E  1580
                                058E  1581 ;move the log message field from the packet into the array ILOGMSG,
                                058E  1582 ;which was specified in the call to XF$SETUP
                                058E  1583 ;R9 contains packet size of device message
                                058E  1584
58 A6   00   20 A749    5A   2C  058E  1585         MOVC5   R10, XF$B_PKT_DEVMSG(R7)[R9], #0, -
                        54 B6           0596
                                0598  1586                 CTX$W_ILOGSIZ(R6), @CTX$L_ILOGMSG(R6)
                                0598  1587 10$:     DEFAULT_TEST    <LOGFLAG/4>, TRANSFER_STATUS, TRANSFER_STATUS
                                05A2  1588                                             ;if LOGFLAG defaulted goto TRANSFER_STAT
                     18 BC   52   92  05A2  1589         MCOMB   R2, @LOGFLAG(AP)        ;set LOGFLAG appropriately
                                05A6  1590
                                05A6  1591 ;++
                                05A6  1592 ; return the third through the eighth longword of the command packet
                                05A6  1593 ; to the user by copying them into CONTXT
                                05A6  1594 ;--
                                05A6  1595 TRANSFER_STATUS:
08 A6   08 A7   7D  05A6  1596         MOVQ    XF$B_PKT_MSGLEN(R7), CTX$L_CONTROL(R6)
                                05AB  1597                                             ;control longword and byte count
10 A6   10 A7   7D  05AB  1598         MOVQ    XF$L_PKT_BFRADR(R7), CTX$L_BFRVA(R6)
                                05B0  1599                                             ;buf addr & residual mem byte count
18 A6   18 A7   7D  05B0  1600         MOVQ    XF$L_PKT_RDBCNT(R7), CTX$L_DDICNT(R6)
                                05B5  1601                                             ;residual DDI count  and
                                05B5  1602                                             ;DR32 status longword
         50   01   3C  05B5  1603         MOVZWL  #SS$_NORMAL, R0                 ;success status
```

```
                                                                I 6
                     05B8  1605 ;++
                     05B8  1606 ;store the status of GETPKT now (if a status argument was given),
                     05B8  1607 ;before the call to the ACTION routine.  GETPKT's status is an input
                     05B8  1608 ;to the ACTION routine.
                     05B8  1609 ;--
                     05B8  1610 STORE_STATUS:
                     05B8  1611         DEFAULT_TEST    <STATUS/4>, 10$, 10$
                     05C2  1612                                         ;if STATUS defaulted goto 10$
           1C BC  50 D0  05C2  1613         MOVL    R0, @STATUS(AP) ;store status
              31  50 E9  05C6  1614 10$:    BLBC    R0, END_GETPKT          ;if no packet, goto end
                     05C9  1615
                     05C9  1616 ;determine size of packet through log message field
                     05C9  1617 ;R9 contains packet size of device message
                     05C9  1618 ;R11 contains packet size of log message field
                     05C9  1619
        53  20 A94B  9E 05C9  1620         MOVAB   XF$B_PKT_DEVMSG(R9)[R11], R3
                     05CE  1621                                         ;R3 <- devmsg size+logmsg size
                     05CE  1622                                         ; + size of fixed part of pkt
                     05CE  1623                                         ;(this is an ADDL, not a MOVA)
                     05CE  1624
                     05CE  1625 ;++
                     05CE  1626 ;IF an ACTION routine is associated with this packet
                     05CE  1627 ;       THEN call it
                     05CE  1628 ;the ACTION routine may substitute its status for GETPKT's status
                     05CE  1629 ;--
                     05CE  1630 ACTION_TEST:
     20 0B A7  02    E1 05CE  1631         BBC     #XF$V_PKT_ACTBIT, XF$B_PKT_PKTCTL(R7), RETURN_SPACE
                     05D3  1632                                         ;if bit is clear, there is no
                     05D3  1633                                         ;ACTION routine associated with
                     05D3  1634                                         ;this packet
                     05D3  1635 ;++
                     05D3  1636 ;R3 contains the size of the packet in bytes, up to and including the
                     05D3  1637 ;log message field.  Add this to the base address of the packet to find
                     05D3  1638 ;the addresses of the ACTION routine and the ACTION
                     05D3  1639 ;routine's parameter.  Then add the size of the two addresses to R3, to
                     05D3  1640 ;calculate the total size of the command packet
                     05D3  1641 ;--
                     05D3  1642
        54    6743  9E 05D3  1643         MOVAB   (R7)[R3], R4            ;R4 <- addr of addr of ACTION
                     05D7  1644                                         ;routine
          53    08  C0 05D7  1645         ADDL2   #8, R3                  ;R3 <- total size of packet
                     05DA  1646
                     05DA  1647
                     05DA  1648 ;input arguments to ACTION routine
            1C AC    DD 05DA  1649         PUSHL   STATUS(AP)
            10 AC    DD 05DD  1650         PUSHL   INDEX(AP)
            0C AC    DD 05E0  1651         PUSHL   FUNC(AP)
            18 AC    DD 05E3  1652         PUSHL   LOGFLAG(AP)
            14 AC    DD 05E6  1653         PUSHL   DEVFLAG(AP)
            04 A4    DD 05E9  1654         PUSHL   4(R4)                   ;addr of ACTION routine param
            04 AC    DD 05EC  1655         PUSHL   CONTXT(AP)
     00 B4    07    FB 05EF  1656         CALLS   #7, @(R4)               ;call user-supplied ACTION
                     05F3  1657                                         ;routine
                     05F3  1658                                         ;status returned in STATUS arg
```

```
                        05F3  1660 ;++
                        05F3  1661 ;return the memory the command packet was built from
                        05F3  1662 ;--
                        05F3  1663 RETURN_SPACE:
                        05F3  1664                                              ;inputs to XFSSDEALOCPKT:
      51    5C A6   DE  05F3  1665          MOVAL   CTXSL_FREELIST(R6), R1      ;R1: addr of freelist header
                        05F7  1666                                              ;R3: size of packet in bytes
                        05F7  1667                                              ;R7: addr of packet to return
            FDD8    30  05F7  1668          BSBW    XFSSDEALOCPKT               ;return the packet space
                        05FA  1669                                              ;(R1, R3, R7 are destroyed)
                        05FA  1670
                        05FA  1671 END_GETPKT:
                     04 05FA  1672          RET
                        05FB  1673
```

```
05FB  1675          .SBTTL  GET_ADDR -- GET PACKET ADDRESS
05FB  1676 ;++
05FB  1677 ; FUNCTIONAL DESCRIPTION:
05FB  1678 ;
05FB  1679 ;       This routine is called by XF$GETPKT to remove a packet from the
05FB  1680 ;       TERMQ and return its address.  The routine, depending on
05FB  1681 ;       conditions,
05FB  1682 ;       (1) returns with address of packet, or
05FB  1683 ;       (2) returns with status "TERMQ empty", or
05FB  1684 ;       (3) determines that this data transaction has completed, and
05FB  1685 ;           calls XF$CLEANUP before returning
05FB  1686 ;
05FB  1687 ; CALLING SEQUENCE:
05FB  1688 ;
05FB  1689 ;       BSBB/W  GET_ADDR
05FB  1690 ;               called by: XF$GETPKT
05FB  1691 ;               calls (under conditions) XF$CLEANUP
05FB  1692 ;
05FB  1693 ; INPUT PARAMETERS:
05FB  1694 ;
05FB  1695 ;       R2 is a switch that determines what action to take when
05FB  1696 ;       TERMQ is empty
05FB  1697 ;               R2 = 0: wait for event flag
05FB  1698 ;               R2 .NE. 0: immediate return with "TERMQ empty" status
05FB  1699 ; IMPLICIT INPUTS:
05FB  1700 ;
05FB  1701 ;       R6 contains the address of the CONTXT array
05FB  1702 ;       fields in CONTXT:
05FB  1703 ;               CTX$L_CMDBLK
05FB  1704 ;               CTX$Q_IOSB
05FB  1705 ;               CTX$W_EFN
05FB  1706 ;
05FB  1707 ; OUTPUT PARAMETERS:
05FB  1708 ;
05FB  1709 ;       R7 contains address of command packet, if one was successfully
05FB  1710 ;       removed from the TERMQ
05FB  1711 ;
05FB  1712 ; IMPLICIT OUTPUTS:
05FB  1713 ;
05FB  1714 ;       NONE
05FB  1715 ;
```

```
;  COMPLETION CODES:
;
;       R0 contains status of call
;       status returns:
;               (1) SS$_NORMAL: normal successful completion
;                                packet address is in R7
;               (2) SS$_BADQUEUEHDR:    interlocked queue timeout
;               (3) SHR$_HALTED:        XFSCLEANUP was called
;               (4) SHR$_QEMPTY: no packet, but transfer still going
;               (5) SHR$_NOCMDMEM:      command memory not allocated at
;                                       the time this routine was called
;               (6) error returns from system calls
;                       $WAITFR
;                       LIB$FREE_VM
;                       LIB$DASSGN
;
;  SIDE EFFECTS:
;
;       If XFSCLEANUP was called (it is called when the TERMQ is empty
;       and the transfer is halted), then the command area was
;       deallocated and the device's channel deassigned.
;
;--
```

```
                                05FB  1741 GET_ADDR:
                     0406 8F    BB  05FB  1742         PUSHR   #^M<R1,R2,R10>
                   5A   24 A6   D0  05FF  1743         MOVL    CTX$L_CMDBLK(R6), R10   ;R10 <- addr of command block
                           63   13  0603  1744         BEQL    CLEANOP_DONE            ;if 0, command area has been
                                0605  1745                                             ;deallocated
                                0605  1746
                                0605  1747 ;++
                                0605  1748 ;attempt to remove packet from head of TERM queue
                                0605  1749 ;if succeed in removing a packet then goto HAVE_PACKET
                                0605  1750 ;this is partly an otpimization to prevent clearing the event flag when
                                0605  1751 ;there is a packet on the TERMQ and partly a test to see if CLEANUP can
                                0605  1752 ;be done .
                                0605  1753 ;--
                                0605  1754 REM_TERMQ:
                           51   D4  0605  1755         CLRL    R1                      ;retry count
                     57    08 AA   5E  0607  1756 10$:  REMQHI  CMD$L_TERMQ(R10), R7    ;R7 <- addr of packet
                           69   1C  060B  1757         BVC     HAVE_PACKET             ;removal succeeded
                           0A   1E  060D  1758         BCC     20$                     ;TERMQ empty
                 F0 51  0000C350 8F    F3  060F  1759         AOBLEQ  #RETRY_LIMIT, R1, 10$   ;queue locked. retry.
                                0617  1760
                                0617  1761 ;exceeded retry limit and queue is still locked
                                0617  1762 ;assume queue can no longer be valid
                                0617  1763
                           56   11  0617  1764         BRB     QUEUE_ERROR
                                0619  1765
                                0619  1766 ;++
                                0619  1767 ;there is no packet on the TERMQ; if in addition the transfer is
                                0619  1768 ;halted, then clean up.
                                0619  1769 ;--
                           66   B5  0619  1770 20$:    TSTW    CTX$Q_IOSB(R6)          ;test status of transfer
                           38   12  061B  1771         BNEQ    CLEANOP                 ;br if transfer halted
                                061D  1772
                                061D  1773 ;++
                                061D  1774 ;come here if there is no packet on the TERMQ but the transfer is still
                                061D  1775 ;going.  Test R2 to determine whether to immediately return with
                                061D  1776 ;"TERMQ empty" status or whether to wait for the event flag to be set.
                                061D  1777 ;--
                           52   95  061D  1778         TSTB    R2                      ;wait for event flag?
                           5A   12  061F  1779         BNEQ    TERMQ_EMPTY             ;no, immediate return
                                0621  1780 ;++
                                0621  1781 ;come here to wait for an event flag to be set before
                                0621  1782 ;re-attempting to remove an entry from the TERM queue
                                0621  1783 ;--
                                0621  1784 WAIT_FOR_EF:
                 52    42 A6   3C  0621  1785         MOVZWL  CTX$W_EFN(R6), R2       ;get event flag number
                                0625  1786         $CLREF_S        EFN = R2        ;clear event flag
```

```
                        51   D4  062E  1788                CLRL    R1                              ;retry count
                     57 08   AA  5E  0630  1789 10$:        REMQHI  CMD$L_TERMQ(R10), R7            ;R7 <- addr of packet
                        40   1C  0634  1790                BVC     HAVE_PACKET                     ;removal succeeded
                        0B   1E  0636  1791                BCC     20$                             ;TERMQ empty
                FO 51 0000C350 8F  F3  0638  1792                AOBLEQ  #RETRY_LIMIT, R1, 10$   ;queue locked. retry.
                              0640  1793
                              0640  1794 ;exceeded retry limit and queue is still locked
                              0640  1795 ;assume queue can no longer be valid
                              0640  1796
                            002C  31  0640  1797                BRW     QUEUE_ERROR
                              0643  1798
                              66  B5  0643  1799 20$:        TSTW    CTX$Q_IOSB(R6)                  ;has transfer halted?
                              0E  12  0645  1800                BNEQ    CLEANUP                         ;yes, go clean up
                              0647  1801
                              0647  1802                $WAITFR_S       EFN = R2                ;wait for flag to be set
                            B2 50  E8  0650  1803                BLBS    R0, REM_TERMQ                   ;re-attempt a packet
                              28  11  0653  1804                BRB     END_GET_ADDR                    ;R0 contains error status from
                              0655  1805                                                              ;WAITFR call
                              0655  1806 ;++
                              0655  1807 ;Come here iff there is nothing on TERMQ and transfer is halted.
                              0655  1808 ;--
                              0655  1809 CLEANUP:
                              56  DD  0655  1810                PUSHL   R6                              ;addr of CONTXT array
                00000685'EF   01  FB  0657  1811                CALLS   #1, XF$CLEANUP                  ;(1) deallocates command area
                              065E  1812                                                              ;(2) deassigns channel
                           1F 50  E9  065E  1813                BLBC    R0, END_GET_ADDR
                      50   1270 8F  3C  0661  1814                MOVZWL  #SHR$_HALTED, R0                ;transfer JUST halted
                              18  11  0666  1815                BRB     END_GET_ADDR
                              0668  1816
                              0668  1817
                              0668  1818 ;status paths
                              0668  1819
                              0668  1820 CLEANUP_DONE:
                50   1278 8F  3C  0668  1821                MOVZWL  #SHR$_NOCMDMEM, R0              ;command area deallocated
                              11  11  066D  1822                BRB     END_GET_ADDR
                              066F  1823
                              066F  1824 QUEUE_ERROR:
                50   0394 8F  3C  066F  1825                MOVZWL  #SS$_BADQUEUEHDR, R0            ;interlock timeout occurred
                              0A  11  0674  1826                BRB     END_GET_ADDR
                              0676  1827
                              0676  1828 HAVE_PACKET:
                     50   01  3C  0676  1829                MOVZWL  #SS$_NORMAL, R0                 ;packet's address is in R7
                              05  11  0679  1830                BRB     END_GET_ADDR
                              067B  1831
                              067B  1832 TERMQ_EMPTY:
                50   1280 8F  3C  067B  1833                MOVZWL  #SHR$_QEMPTY, R0                ;no packet on TERMQ
                              0680  1834
                              0680  1835 END_GET_ADDR:
                              0680  1836
                     0406 8F  BA  0680  1837                POPR    #^M<R1,R2,R10>
                              05  0684  1838                RSB
                              0685  1839
```

```
                        0685  1841          .SBTTL  XF$CLEANUP
                        0685  1842 ;++
                        0685  1843 ; FUNCTIONAL DESCRIPTION:
                        0685  1844 ;
                        0685  1845 ;       (1) deassign channel
                        0685  1846 ;       (2) deallocate virtual memory
                        0685  1847 ;
                        0685  1848 ; CALLING SEQUENCE:
                        0685  1849 ;
                        0685  1850 ;       CALLS/G  XF$CLEANUP(CONTXT, [STATUS])
                        0685  1851 ;
                        0685  1852 ; INPUT PARAMETERS:
                        0685  1853 ;
                00000004 0685  1854          CONTXT = 4
                        0685  1855 ;
                        0685  1856 ; IMPLICIT INPUTS:
                        0685  1857 ;
                        0685  1858 ;       fields in CONTXT array:
                        0685  1859 ;               CTX$L_CMDBLK
                        0685  1860 ;               CTX$L_CMDSIZ
                        0685  1861 ;
                        0685  1862 ; OUTPUT PARAMETERS:
                        0685  1863 ;
                00000008 0685  1864          STATUS = 8             ;optional status word
                        0685  1865 ;
                        0685  1866 ; IMPLICIT OUTPUTS:
                        0685  1867 ;
                        0685  1868 ;       R0 contains status also (used when XF$GETPKT calls XF$CLEANUP)
                        0685  1869 ;
                        0685  1870 ; COMPLETION CODES:
                        0685  1871 ;
                        0685  1872 ;       SS$_NORMAL      -- successful completion
                        0685  1873 ;       SHR$_NOCMDMEM   -- command memory was not allocated at the time
                        0685  1874 ;                          of this call to XF$CLEANUP
                        0685  1875 ;       error returns from:
                        0685  1876 ;               LIB$FREE_VM
                        0685  1877 ;               $DASSGN
                        0685  1878 ;
                        0685  1879 ; SIDE EFFECTS:
                        0685  1880 ;
                        0685  1881 ;       NONE
                        0685  1882 ;
                        0685  1883 ;--
```

```
                                0685  1885
                         0044   0685  1886          .ENTRY  XF$CLEANUP      ^M<R2,R6>
                                0687  1887
          56    04 AC    D0    0687  1888          MOVL    CONTXT(AP), R6          ;R6 <- addr of CONTXT
                                068B  1889
          50  1278 8F    3C    068B  1890          MOVZWL  #SHR$_NOCMDMEM, R0      ;assume cmd memory not allocated
                24 A6    D5    0690  1891          TSTL    CTX$L_CMDBLK(R6)        ;is address non-zero?
                   28    13    0693  1892          BEQL    10$                     ;branch if cmd mem not allocated
                                0695  1893
                                0695  1894 ;deassign channel (also cancels any IO still in progress)
                                0695  1895
     52  00000000'EF    DE    0695  1896          MOVAL   DEVICE_FAB, R2          ;channel number still in FAB
                                069C  1897          $DASSGN_S       CHAN = FAB$L_STV(R2)
                                06A7  1898                                          ;deassign the channel
                13 50    E9    06A7  1899          BLBC    R0, 10$                 ;error from $DASSGN
                                06AA  1900
                                06AA  1901 ;deallocate dynamic virtual memory
                                06AA  1902
                24 A6    DF    06AA  1903          PUSHAL  CTX$L_CMDBLK(R6)        ;address of virtual memory
                20 A6    DF    06AD  1904          PUSHAL  CTX$L_CMDSIZ(R6)        ;size of virtual memory block
00000000'GF       02    FB    06B0  1905          CALLS   #2, G^LIB$FREE_VM       ;return the memory
                03 50    E9    06B7  1906          BLBC    R0, 10$                 ;error return
                24 A6    D4    06BA  1907          CLRL    CTX$L_CMDBLK(R6)        ;signal command mem returned
                                06BD  1908
                                06BD  1909 ;see if STATUS argument supplied
                                06BD  1910
                                06BD  1911 10$:     DEFAULT_TEST    <STATUS/4>, END_CLEANUP, END_CLEANUP
          08 BC    50    D0    06C7  1912          MOVL    R0, @STATUS(AP) ;store status of call
                                06CB  1913
                                06CB  1914 END_CLEANUP:
                          04    06CB  1915          RET
                                06CC  1916
                                06CC  1917          .END
```

```
$$.TAB                  = 00000000 R      03
$$.TABEND               = 00000050 R      03
$$.TMP                  = 00000000
$$.TMP1                 = 00000001
$$.TMP2                 = 000000CF
$$T1                    = 00000000
ACTION                  = 00000010
ACTION_FIELD              000002F8 R      02
ACTION_ROUTINE            00000255 R      02
ACTION_TEST               000005CE R      02
ACTPARM                 = 00000014
ALOC                      00000092 R      02
ALOCCMD                   000000A8 R      02
ALOCMASK                = 0000001F
ANOTHER_PKT               0000048A R      02
ASSIGN_CHN                0000015B R      02
ASTPARM                 = 00000010
A_OK                      000004CE R      02
BADPARM                   000001BF R      02
BAD_QUEUE                 000004D6 R      02
BARRAY                  = 00000008
BITS                      00000262 R      02
BR                        0000024F R      02
BUFSIZ                  = 0000000C
CLEANUP                   00000655 R      02
CLEANUP_DONE              00000668 R      02
CMD$L_FREEQ             = 00000010
CMD$L_INPTQ             = 00000000
CMD$L_TERMQ             = 00000008
CMDSIZ                  = 00000024
CMDSIZ_K                = 00000003
CMDSIZ_TEST               0000006C R      02
COMSIZ                    0000007E R      02
CONTXT                  = 00000004
CRITICAL_BIT            = 00000000
CRITICAL_MASK           = 00000001
CTX$B_CMDTBL            = 00000020
CTX$B_CMTFLAGS            00000039
CTX$B_DATART              00000038
CTX$L_ASTPARM             00000048
CTX$L_BFRVA               00000010
CTX$L_BUFSIZ              0000004C
CTX$L_BYTECNT             0000000C
CTX$L_CMDBLK              00000024
CTX$L_CMDSIZ              00000020
CTX$L_CONTROL             00000008
CTX$L_DATABLK             0000002C
CTX$L_DATASIZ             00000028
CTX$L_DDICNT              00000018
CTX$L_DSL                 0000001C
CTX$L_FREELIST            0000005C
CTX$L_GOBITADR            0000003C
CTX$L_IDEVMSG             00000050
CTX$L_ILOGMSG             00000054
CTX$L_MEMCNT              00000014
CTX$L_PKTAST              00000044
CTX$L_PRE_AST             00000030
CTX$L_PRE_PARM            00000034
CTX$Q_IOSB                00000000
CTX$W_EFN                 00000042
CTX$W_IDEVSIZ             0000005A
CTX$W_ILOGSIZ             00000058
CTX$W_NUMBUF              00000C040
DATART                  = 0000001C
DATART_DEF              = 00000000
DATART_TEST               00000148 R      02
DEALLOCATE                00000368 R      02
DEVFLAG                 = 00000014
DEVFLAG_TEST              0000055B R      02
DEVICE_FAB                00000000 R      03
DEVMSG                  = 00000014
DEVNAM                  = 00000008
DEVSIZ                  = 00000018
DIFSIZE                 = 00000010
DISSECT_PKT               00000517 R      02
DUMMY_ADR               = 00000200
EFN                     = 00000014
EFN_DEF                 = 00000015
EFN_TEST                  0000012A R      02
END                       000000A7 R      02
END_ALOCPKT               000003CF R      02
END_CLEANUP               000006CB R      02
END_DEALOCPKT             00000425 R      02
END_FREESET               000004E9 R      02
END_GETPKT                000005FA R      02
END_GET_ADDR              00000680 R      02
END_PKTBLD                0000038D R      02
END_PRE_AST               00000207 R      02
END_STARTDEV              000001D0 R      02
EQUAL                     000003BB R      02
FAB$B_FNS               = 00000034
FAB$C_BID               = 00000003
FAB$C_BLN               = 00000050
FAB$C_SEQ               = 00000000
FAB$C_VAR               = 00000002
FAB$L_ALQ               = 00000010
FAB$L_FNA               = 0000002C
FAB$L_FOP               = 00000004
FAB$L_STV               = 0000000C
FAB$V_CHAN_MODE         = 00000002
FAB$V_FILE_MODE         = 00000004
FAB$V_LNM_MODE          = 00000000
FAB$V_UFO               = 00000011
FAB$W_GBC               = 00000048
FIELDS_DONE               00000344 R      02
FIND_SIZE                 00000437 R      02
FINISH                    00000099 R      02
FUNC                    = 0000000C
FUNC_FIELD                000002A0 R      02
GET_ADDR                  000005FB R      02
GO                        00000252 R      02
GRANULARITY             = 00000007
HAVE_PACKET               00000676 R      02
IDEVMSG                 = 00000014
```

```
IDEVSIZ                 =  00000018
ILOGMSG                 =  0000001C
ILOGSIZ                 =  00000020
IMMEDIATE_EXIT             000001F0 R      02
INDEX                   =  00000010
INDEX_FIELD                000002BD R      02
INDEX_TEST                 0000053C R      02
INSERT_AT_HEAD             00000333 R      02
INSERT_AT_TAIL             00000344 R      02
INTCTRL                 =  0000000C
INT_DEFAULT             =  00000000
INV                        000002B5 R      02
INVALID_ARG                0000035E R      02
IO$M_SETEVF             =  00000040
IO$_STARTDATA           =  00000038
LIB$FREE_VM                ********   X    02
LIB$GET_VM                 ********   X    02
LOGFLAG                 =  00000018
LOGFLAG_TEST               00000580 R      02
LOGSIZ                  =  0000001C
LOGSIZE                    00000233 R      02
MODES                   =  00000020
MODES_DEFAULT           =  00000000
MODES_FIELD                0000031B R      02
MODE_TEST                  00000139 R      02
MSG_ARRAYS                 0000002A R      02
NEXT                       00000282 R      02
NOT_MEM                    000004DD R      02
NO_MEM                     00000373 R      02
NUMBUF                  =  00000010
NUMPKT                  =  00000008
OK                         000002B8 R      02
PAGEMASK                =  000001FF
PKTAST                  =  0000000C
PKTAST_TEST                000000FF R      02
PRE_AST                    000001D1 R      02
QUADWORD_MASK           =  00000007
QUEUE_ERROR                0000066F R      02
Q_FAILURE                  00000363 R      02
REM_TERMQ                  00000605 R      02
RETRY_LIMIT             =  0000C350
RETURN_SPACE               000005F3 R      02
SET_GO_BIT                 00000355 R      02
SHR$_HALTED             =  00001270
SHR$_NOCMDMEM           =  00001278
SHR$_QEMPTY             =  00001280
SS$_BADPARAM            =  00000014
SS$_BADQUEUEHDR         =  00000394
SS$_INSFMEM             =  00000124
SS$_NORMAL              =  00000001
STAT                       000001C2 R      02
STATUS                  =  00000008
STORE_STAT                 0000037F R      02
STORE_STATUS               000005B8 R      02
SYS$CLREF                  ********  GX    02
SYS$CREATE                 ********  GX    02
SYS$DASSGN                 ********  GX    02
SYS$DCLAST                 ********  GX    02
SYS$QIO                    ********  GX    02
SYS$SETAST                 ********  GX    02
SYS$WAITFR                 ********  GX    02
TERMQ_EMPTY                0000067B R      02
TRANSFER_HALTED            0000037A R      02
TRANSFER_STATUS            000005A6 R      02
TRANS_HALTED               000004E4 R      02
TRUE                                =  000000FF
WAITFLG                             =  00000008
WAIT_FOR_EF                00000621 R      02
XF$$ALOCPKT                0000038E RG     02
XF$$DEALOCPKT              000003D2 RG     02
XF$B_CMT_FLAGS                      =  00000019
XF$B_CMT_RATE                       =  00000018
XF$B_PKT_CMDCTL                     =  0000000A
XF$B_PKT_DEVMSG                     =  00000020
XF$B_PKT_LOGLEN                     =  00000009
XF$B_PKT_MSGLEN                     =  00000008
XF$B_PKT_PKTCTL                     =  0000000B
XF$CLEANUP                 00000685 RG     02
XF$FREESET                 00000428 RG     02
XF$GETPKT                  000004F8 RG     02
XF$K_CMT_LENGTH                     =  00000020
XF$L_CMT_GBITAD                     =  0000001C
XF$L_CMT_PASTAD                     =  00000010
XF$L_CMT_PASTPM                     =  00000014
XF$L_PKT_BFRADR                     =  00000010
XF$L_PKT_BFRSIZ                     =  0000000C
XF$L_PKT_DSL                        =  0000001C
XF$L_PKT_RDBCNT                     =  00000018
XF$M_CMT_SETRTE                     =  00000001
XF$M_PKT_ACTBIT                     =  00000004
XF$PRTBLD                  00000208 RG     02
XF$SETUP                   00000000 RG     02
XF$STARTDEV                000000E7 RG     02
XF$S_PKT_INTCTL                     =  00000002
XF$V_PKT_ACTBIT                     =  00000002
XF$V_PKT_DDISTS                     =  00000010
XF$V_PKT_FREQPK                     =  00000003
XF$V_PKT_HT                         =  00000008
XF$V_PKT_INTCTL                     =  00000006
XF$V_PKT_LOG                        =  00000006
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000060 (96.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_XF$CODE	000006CC (1740.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
_XF$DATA	00000050 (80.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.10	00:00:00.52
Command processing	157	00:00:00.58	00:00:01.96
Pass 1	396	00:00:15.45	00:00:31.25
Symbol table sort	0	00:00:02.10	00:00:04.31
Pass 2	327	00:00:04.87	00:00:09.06
Symbol table output	27	00:00:00.20	00:00:01.01
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	945	00:00:23.33	00:00:48.15

The working set limit was 1950 pages.
90665 bytes (178 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1377 non-local and 33 local symbols.
1917 source lines were read in Pass 1, producing 31 object records in Pass 2.
34 pages of virtual memory were used to define 29 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[IOSUP.SRC]DRDEF.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	22
TOTALS (all libraries)	24

1597 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

MACRO/DISABLE=TRACE/LIS=LIS$:DRSUP/OBJ=OBJ$:DRSUP MSRC$:DRSUP/UPDATE=(ENH$:DRSUP)+SRC$:DRDEF/LIB

LASWEEP
LIS

XFDEF
FOR

LASNDLDRQ
LIS

LABUFFER
LIS

JOBCTL

JOBCTL
MAP

JOBCTLDEF
REQ

DRSUP
LIS

SYSQUEDEF
SDL

IOSUP

ACCOUNTNG
LIS

DRDEF
MAR